Docket No. AUS920030412US1 *PATENT*

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: **Brown et al.**	§	
	§	Group Art Unit: **2152**
Serial No. **10/614,629**	§	
	§	Examiner: **Hussain, Tauqir**
Filed: **July 3, 2003**	§	
	§	Confirmation No.: **7818**
For: **Method and Apparatus for Managing a Remote Data Processing System**	§	

35525

PATENT TRADEMARK OFFICE
CUSTOMER NUMBER

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF (37 C.F.R. 41.37)

This brief is in furtherance of the Notice of Appeal, filed in this case on May 27, 2008.

A fee of $510.00 is required for filing an Appeal Brief. Please charge this fee to IBM Corporation Deposit Account No. 09-0447. No additional fees are believed to be necessary. If, however, any additional fees are required, I authorize the Commissioner to charge these fees which may be required to IBM Corporation Deposit Account No. 09-0447. No extension of time is believed to be necessary. If, however, an extension of time is required, the extension is requested, and I authorize the Commissioner to charge any fees for this extension to IBM Corporation Deposit Account No. 09-0447.

REAL PARTY IN INTEREST

The real party in interest in this appeal is the following party: International Business Machines Corporation of Armonk, New York.

RELATED APPEALS AND INTERFERENCES

This appeal has no related proceedings or interferences.

STATUS OF CLAIMS

A. TOTAL NUMBER OF CLAIMS IN APPLICATION

The claims in the application are: 1-21

B. STATUS OF ALL THE CLAIMS IN APPLICATION

Claims canceled: None

Claims withdrawn from consideration but not canceled: None

Claims pending: 1-21

Claims allowed: None

Claims rejected: 1-21

Claims objected to: None

C. CLAIMS ON APPEAL

The claims on appeal are: 1-21

STATUS OF AMENDMENTS

An After Final Office Action amendment was filed on May 27, 2008. An Advisory Action dated June 13, 2008 indicated that the amendment will not be entered.

SUMMARY OF CLAIMED SUBJECT MATTER

A. CLAIM 1 - INDEPENDENT

The subject matter of Claim 1 is directed to a method in a data processing system for providing host information (Specification, page 11, line 29 – page 12, line 4). The method includes receiving a request for host information for a remote computer from a requestor, where the request includes one of a host name or an Internet Protocol address and is received from the requestor (Specification, page 13, lines 17-25; page 18, lines 20-22; Figure 10, block 1004). The method includes identifying a media access control address and a subnet mask using the request (Specification, page 13, lines 26-28; page 18, lines 22-23). The method includes returning a response to the requestor, where the response includes the media access control address and the subnet mask (Specification, page 13, lines 26-28; page 18, lines 22-23).

B. CLAIM 8 - INDEPENDENT

The subject matter of claim 8 is directed to a data processing system for providing host information (Specification, page 11, line 29 – page 12, line 4). The data processing system includes receiving means for receiving, by the data processing system, a request for host information for a remote computer from a requestor wherein the request includes one of a host name or an Internet Protocol address and is received from the requestor (Specification, page 13, lines 17-25; page 18, lines 20-22; Figure 10, block 1004). The data processing system includes identifying means for identifying, by the data processing system, a media access control address and a subnet mask for the remote computer using the request (Specification, page 13, lines 26-28; page 18, lines 22-23). The data processing system includes returning means for returning, by the data processing system, a response to the requestor, wherein the response includes the media access control address and the subnet mask for the remote computer (Specification, page 13, lines 26-28; page 18, lines 22-23).

The structure corresponding to the receiving means, identifying means and returning means is depicted at element 408 of Figure 4, as described in the Specification at page 13, lines 17-30.

C. CLAIM 15 - INDEPENDENT

The subject matter of Claim 15 is directed to a computer program product encoded in a computer readable storage medium and operable in a data processing system for providing host information (Specification, page 11, line 29 – page 12, line 4). The computer program product includes first instructions for receiving, by the data processing system, a request for host information for a remote computer from a requestor wherein the request includes one of a host name or an Internet Protocol address and is received from the requestor (Specification, page 13, lines 17-25; page 18, lines 20-22; Figure 10, block 1004). The computer program product includes second instructions for identifying, by the data processing system, a media access control address and a subnet mask for the remote computer using the request (Specification, page 13, lines 26-28; page 18, lines 22-23). The computer program product includes third instructions for returning, by the data processing system, a response to the requestor, wherein the response includes the media access control address and the subnet mask for the remote computer (Specification, page 13, lines 26-28; page 18, lines 22-23).

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

The grounds of rejection to review on appeal are as follows:

A. GROUND OF REJECTION 1

Whether Claims 1, 8 and 15 were properly rejected under 35 U.S.C. § 102 as being anticipated by Hutchison et al. (U.S. Patent No. 7,249,191 B1), hereinafter "Hutchison";

B. GROUND OF REJECTION 2

Whether Claims 2, 9 and 16 were properly rejected as being obvious over Hutchison et al. (U.S. Patent No. 7,249,191 B1), hereinafter "Hutchison" in view of Bullman et al. (U.S. Publication No. 2002/0162038 A1), hereinafter "Bullman", under 35 U.S.C. § 103;

C. GROUND OF REJECTION 3

Whether Claims 3, 6, 7, 10, 13, 14, 17 and 20 were properly rejected as being obvious over Hutchison et al. (U.S. Patent No. 7,249,191 B1), hereinafter "Hutchison" in view of Harrison et al. (U.S. Publication No. 2004/0177133 A1), hereinafter "Harrison", under 35 U.S.C. § 103;

D. GROUND OF REJECTION 4

Whether Claims 4, 11 and 18 were properly rejected as being obvious over Hutchison et al. (U.S. Patent No. 7,249,191 B1), hereinafter "Hutchison" in view of Matsuda et al. (U.S. Patent No. 7,039,688 B2), hereinafter "Matsuda", under 35 U.S.C. § 103; and

E. GROUND OF REJECTION 5

Whether Claims 5, 12, 19 and 21 were properly rejected as being obvious over Hutchison et al. (U.S. Patent No. 7,249,191 B1), hereinafter "Hutchison" in view of Bahl. (U.S. Patent No. 6,957,276 B1), hereinafter "Bahl", under 35 U.S.C. § 103.

ARGUMENT

A. GROUND OF REJECTION 1 (Claims 1, 8 and 15)

Claims 1, 8 and 15 stand rejected under 35 U.S.C. § 102 as being anticipated by Hutchison et al. (U.S. Patent No. 7,249,191 B1), hereinafter "Hutchison".

1. *Claim 1*

For a prior art reference to anticipate in terms of 35 U.S.C. 102, every element of the claimed invention must be identically shown in a single reference. *In re Bond*, 910 F.2d 831, 15 USPQ2d 1566 (Fed. Cir. 1990). "To establish inherency," as stated by the Federal Circuit, "the extrinsic evidence `must make clear that the missing descriptive matter is necessarily present in the thing described in the reference, and that it would be so recognized by persons of ordinary skill.'" *In re Robertson*, 169 F.3d 743, 745 [49 USPQ2d 1949] (Fed. Cir. 1999); see also *Continental Can Co. U.S.A., Inc. v. Monsanto Co.*, 948 F.2d 1264, 1268 [20 USPQ2d 1746] (Fed. Cir. 1991). Such inherency may not be established by "`probabilities or possibilities.'" *Continental Can*, 948 F.2d at 1269 (quoting *In re Oelrich*, 666 F.2d 578, 581 [212 USPQ 323] (C.C.P.A. 1981)).

With respect to Claim 1 (and similarly for Claims 8 and 15), such claim recites "receiving a request for host information for a remote computer from a requestor wherein the request includes one of a host name or an Internet Protocol address and is received from the requestor", "identifying a media access control address *and a subnet mask* using the request" and "returning a response to the requestor, wherein the response includes the media access control address *and the subnet mask*" (emphasis added by Appellants). The cited reference does not teach specific steps of (1) identifying a *subnet mask* using a request that was received, or (2) **returning a response that includes a *subnet mask***.

In rejecting Claim 1, the Examiner states that it is inherent 'that subnet mask will be there along with MAC address or else the packet will be loss (sic) not knowing which network or which segment of the network it belongs to'. Appellants urge clear error in such inherency assertion, as one of the fundamental premises that the teachings of the cited reference are directed to is to provide a transparent platform that intercepts a file from a source across a first IP

connection, and resends such file over a second TCP connection (Hutchinson Abstract). This redirection of the file from the first IP connection to a second IP connection is done using a specialized network address translation bridge. Importantly, this bridge 'spoofs' a client that wishes to retrieve a particular web page, by using an intervening gateway that intercepts the client's request for a MAC address of the device containing the desired web page, and *this gateway instead provides the client with its own MAC address* (col. 4, lines 18-23; Figure 8, element ARP REPLY 10.0.0.1 = ABCD).

Even more importantly, this intervening gateway is in the *same subnet* as the client (gateway has IP address of 10.0.0.1 and client has IP address of 10.0.0.2 as clearly shown in Figure 8) so there is no issue with losing a packet by not specifying a subnet mask in response to the request for a MAC address, as the gateway 'spoofs' the client by providing *its own* MAC address *which is on the same subnet as the client.* Thus, there is no reason or need for this spoofing gateway to also include a subnet mask in the response to the request for a MAC address, as both devices (gateway and client) are on the same subnet and thus such mask is not needed to ensure proper addressing between the client and the spoofing gateway. Therefore, the extrinsic evidence does not make it clear that the missing claimed feature (subnet mask being returning in a request) is necessarily present in the thing described in the reference, and that it would be so recognized by persons of ordinary skill in the art, as required by *In re Robertson,* supra.

Quite simply, inclusion of such a subnet mask (as claimed) would needlessly consume valuable system bandwidth by transmitting useless information that is not needed – somewhat akin to including a country code when dialing a local, non-international phone call, it's just not done because it is not needed. Therefore, whether or not Hutchinson provides a subnet mask is pure conjecture and speculation. As previously pointed out, inherency may not be established by "probabilities or possibilities" *Continental Can*, supra (quoting *In re Oelrich*, supra).

The Examiner also opines that a subnet mask must be included in the response or else the packet would get lost. Appellants urge that this simply is not so, as can clearly be seen by the SYN ACK response packet shown at 64 of Figure 8 (and as described by Hutchinson at col. 4, lines 46-50) where (1) the source node is fully specified (123.0.0.1:80), (2) the destination node is fully specified (10.0.0.2:900), (3) the source MAC address is fully specified (ABCD), and (4) the destination MAC address is fully specified (EFGH). Because both the source and destination nodes and source and destination MAC addresses are fully specified in this response, there is no

worry regarding lost packets as this SYN ACK response includes all information needed to unambiguously find/address the source and destination devices without needing to use a subnet mask – further evidencing that the inclusion of a subnet mask in such a response (SYN ACK) to a request (SYN) is not inherent.

Rebuttal Response:

In the most recent Response to Arguments provided by the Examiner in their Final Rejection dated February 26, 2008, the Examiner states in rebuttal:

> "it is well-known in the art that in IP-network it is inherent to have a subnet mask … associated with each IP address and since an IP address is assigned to a network card when each network card inherently has a MAC address burned to the hardware itself and therefore any time network card receives the packet or sends the packet as showed in Hutchinson, Fig. 13, all three information goes along together".

Appellants respectfully submit, and as will be shown in detail with documentary evidence provided as attachments hereto, that it is not well-known or inherent to **send** all three of IP address, MAC address *and subnet mask* in a data packet that is transmitted over an IP-network.

Subnets are used to partition a host address space by assigning subnet numbers to the individually partitioned LANs (see, e.g., "RFC 917" attached hereto as Attachment A). There are two parts to a logical IP address – a network portion and a node portion. An internally maintained subnet mask is used by a router or gateway that interconnects a local/private, partitioned network with the public network in order to translate a local address to a global/public address, where the network bits portion of the logical IP address are represented in such address at the bit locations where the subnet mask has 1s in them, and the node bits portion of the logical IP address are represented in such IP address at the bit locations where the subnet mask has 0s in them. Therefore, the subnet mask is used to define which bits of the logical IP address are associated with the network-portion, and which bit of the logical IP address are associated with the node-portion. Thus, for example, when a subnet mask of 255.255.255.0

exists for a given subnet, the upper twenty-four bits of the IP address specify the network-portion of the address (one common address being 192.168.0.x) and the lower eight bits specify a particular device/node on this private subnet network (0-256 are typical values) (see, e,g, "Subnetting" attached hereto as Attachment B).

A typical TCP/IP data packet, such as described by the cited reference, includes both a source IP address as well as a destination IP address, but does not include subnet mask information (see, e.g., "TCP/IP Suite attached hereto as Attachment C, and specifically note the figure on page 1). Importantly, because the router/gateway provides address translation from the local/private address to the public address (see, e.g., "Network Address Translation" attached hereto as Attachment D), *there is no need or reason to include a subnet mask in normal data packets as such information is internally stored within the router that performs the on-the-fly address translation*. The fundamental reason why this is so is that the subnet mask is specific to the local/private networks and is not typically used by *other* local/private networks as the address translation is transparently performed by a router/gateway that uses such network mask that is internally maintained within the router.[1]

However, there is a special circumstance where the subnet mask of one local/private network needs to be known by another local/private network. This special circumstance is a wake-up packet, where a node/device on one local/private network wants/needs to wake-up a node/device on another local/private network. The implementation of the wakeup packet protocol requires specification of the subnet mask in order to awaken a device (Specification page 2, lines 19-30). However, this requires that the system administrator *already know* these requisite subnet masks in order to formulate the wake-up packet (Specification page 2, line 30 – page 3, line 4). One of the fundamental premises of the present invention is to facilitate the automated acquisition of such a subnet mask in order to mitigate these system administrator issues (Specification page 3, lines 5-8). Accordingly, as an expressly provisioned claimed feature, a response is returned to a requestor of host information, and *this response <u>includes the subnet mask</u>* (Appellants' Claim 1) – thereby advantageously allowing the requester to subsequently perform the special wake-up of a device on another network without having to

[1] This is somewhat akin to telephone systems in towns/cities having but a single area code, where the area code is not used when dialing other numbers within the same area code (local call), but the area code is used when dialing other numbers outside the same area code (long distance call).

know up-front what the subnet mask is for this another network.

The cited Hutchinson passage at col. 4, lines 16-21 and Figure 7 describes the following:

> "Client 104 then sends out an address resolution protocol (ARP) request containing the IP address (10.0.0.1) of a gateway 116 asking for the corresponding 48-bit Ethernet hardware address, referred to as destination MAC address. Gateway 116 responds with its MAC address, in this example ABCD."

As can be seen, this passage describes a request that includes an IP address (with such IP address having been described in detail above, which is not a subnet mask), and in response a MAC address is returned (a MAC address, as commonly known to those of ordinary skill in the art, is a physical (as opposed to logical) address of a particular node/device, which is not a subnet mask). This client/gateway exchange is *internal to the private network*, so this request/response exchange *has no need to use subnet masks as a part of such exchange*. Even if subnet masks where used internal to the router in servicing this request (which Appellants deny), there would still be no reason to include a subnet mask *in this response* as the client is requesting an *actual physical hardware address for the gateway* - which is different from an IP address and associated subnet mask (as described in detail above) - that does not use/require a subnet mask. Thus, it is not inherent that a response returned to a client that is requesting a physical hardware address of a router/gateway includes a subnet mask as it is not needed or desired by the client to address the intra-subnetwork-attached router/gateway. Quite simply, while it may be inherent for a router/gateway to *use* a subnet mask (in performing network address translation between two different networks), it is *not* inherent – or even needed or desired – to *include a subnet mask* in a general purpose IP datagram/packet.

Thus, it is urged that Claim 1 is not anticipated by the cited reference as every element recited in such claim is not identically shown in a single reference, per *In re Bond*, *supra*.

2. *Claims 8 and 15*

Appellants initially urge error in the rejection of Claims 8 and 15 for similar reasons to those given above with respect to Claim 1. It should be further noted that per the features of Claims 8 and 15, the data processing system returns a MAC address for a *different computer* (remote computer), and not a MAC address for itself (data processing system), as taught by the

cited reference. Even assuming arguendo that it is inherent that Hutchinson's gateway returns its own subnet mask (which Appellants deny, for the numerous reasons articulated above), there would be no reason for Hutchinson to return a subnet mask for another device ("remote computer"), as per the features of Claims 8 and 15. Therefore, it is further urged that Claims 8 and 15 have been erroneously rejected as every element recited therein is not identically shown in a single reference.

B. GROUND OF REJECTION 2 (Claims 2, 9 and 16)

Claims 2, 9 and 16 stand rejected under 35 U.S.C. § 103 as being unpatentable over Hutchison et al. (U.S. Patent No. 7,249,191 B1), hereinafter "Hutchison" in view of Bullman et al. (U.S. Publication No. 2002/0162038 A1), hereinafter "Bullman".

In determining obviousness, the scope and content of the prior art are... determined; differences between the prior art and the claims at issue are... ascertained; and the level of ordinary skill in the pertinent art resolved. Against this background the obviousness or non-obviousness of the subject matter is determined. *Graham v. John Deere Co.*, 383 U.S. 1 (1966). "Often, it will be necessary for a court to look to interrelated teachings of multiple patents; the effects of demands known to the design community or present in the marketplace; and the background knowledge possessed by a person having ordinary skill in the art, all in order to determine whether there was an apparent reason to combine the known elements in the fashion claimed by the patent at issue." *KSR Int'l. Co. v. Teleflex*, Inc., No. 04-1350 (U.S. Apr. 30, 2007). "Rejections on obviousness grounds cannot be sustained by mere conclusory statements; instead, there must be some articulated reasoning with some rational underpinning to support the legal conclusion of obviousness. *Id.* (citing *In re Kahn*, 441 F.3d 977, 988 (CA Fed. 2006))."

1. *Claims 2, 9 and 16*

Appellants initially urge error in the rejection of Claim 2 (and similarly for Claims 9 and 16) for reasons given above with respect to Claim 1 (of which Claim 2 depends upon), and urge that the additional cited Bullman reference does not overcome the teaching deficiencies identified above with respect to Claim 1.

Further with respect to Claim 2 (and similarly for Claims 9 and 16), it is urged that none of the cited references teach or suggest the claimed feature of "wherein the *requestor* generates a

wake-up packet using the host information and sends the wake-up packet to the remote computer" (emphasis added), with the *requestor* being *the same requestor for which a request for host information was received from*, and where such request (that is received from the requester) includes one of a host name or an Internet Protocol address (per Claim 1). In contrast, per the teachings of Bullman (which is being cited as teaching all features of Claim 2), the PHY device sends a wake-up packet. However, this PHY device is not equivalent to the claimed requestor, since requests that include one of a host name or an Internet Protocol address are not received from this PHY device. Per the features of Claim 2 in combination with Claim 1, *the requestor that generates that wake-up packet using host information is the same requestor that requested the host information*. The combined teachings of the cited reference do not establish a teaching/suggestion of a *same device* that requested host information *also* generates a wake-up packet *using this same (requested) host information*. Therefore, it is further urged that Claim 2 (and similarly for Claims 9 and 16) is not obvious in view of the cited references, and therefore has been erroneously rejected.

C. GROUND OF REJECTION 3 (Claims 3, 6, 7, 10, 13, 14, 17 and 20)

Claims 3, 6, 7, 10, 13, 14, 17 and 20 stand rejected under 35 U.S.C. § 103 as being unpatentable over Hutchison et al. (U.S. Patent No. 7,249,191 B1), hereinafter "Hutchison" in view of Harrison et al. (U.S. Publication No. 2004/0177133 A1), hereinafter "Harrison".

1. *Claims 3, 10 and 17*

Appellants initially urge error in the rejection of Claim 3 (and similarly for Claims 10 and 17) for reasons given above with respect to Claim 1 (of which Claim 3 depends upon), and urge that the additional cited Harrison reference does not overcome the teaching deficiencies identified above with respect to Claim 1.

Further with respect to Claim 3, two different items are received from a dynamic host configuration protocol server - (1) a media access control address and (2) a subnet mask. The things that are received from the DHCP host per the cited Harrison reference are 'host configuration parameters, including an IP address for the client', 'an IP address of a DNS server', 'an IP address of a TFTP server', and 'a name of a device configuration boot file'. There is no teaching or suggestion of any type of *subnet mask being returned by a DHCP server*.

Accordingly, it is further urged that Claim 3 (and similarly for Claims 10 and 17) is not obvious in view of the cited references do to these numerous additional missing claimed features, as described above, and therefore has been erroneously rejected.

2. *Claims 6, 13 and 20*

Appellants initially urge error in the rejection of Claim 6 (and similarly for Claims 13 and 20) for reasons given above with respect to Claim 1 (of which Claim 6 depends upon), and urge that the additional cited Harrison reference does not overcome the numerous teaching deficiencies identified above with respect to Claim 1.

Further with respect to Claim 6 (and similarly for Claims 13 and 20), it is urged that none of the cited references teach or suggest the claimed feature of "wherein the data processing system is a domain name server". It should be noted that Claim 6 should not be interpreted in the abstract as merely reciting a generic domain name server. Rather, Claim 6 must be interpreted in the context of Claim 1 (since Claim 6 depends upon Claim 1). Per the features of Claim 6 when interpreted in the context of Claim 1, such claim recites a domain name server that performs each of the three steps of (1) 'receiving' (a request for host information from a requestor), (2) 'identifying' (a media access control address and a subnet mask), and (3) 'returning' (a response to the requestor). The DNS server as described by the cited Hutchinson passage at col. 4, lines 13-15 does not perform all three of these claimed steps. Rather, such a DNS server is described as being 'somewhere on the internet' that 'returns the 32-bit IP address for WebPages.com'. This DNS server could *not* perform a step of 'identifying a media access control address and a subnet mask using the request' as *it does not have access to this type of information.* The present invention and associated specification description is what is the enabling technology that allows a DNS server to identify this type of information. Prior to the present invention, DNS servers did not have access to media access control addresses and a subnet masks for devices. Thus, the combined teachings of the cited references do not in fact teach or suggest *a <u>DNS server</u> that performs steps of "receiving a request for host information for a remote computer from a requestor wherein the request includes one of a host name or an Internet Protocol address and is received from the requestor", "identifying a media access control address and a subnet mask using the request" and "returning a response to the requestor, wherein the response includes the media access control address and the subnet mask"*, as per the features of Claim 6 in

combination with Claim 1. Accordingly, it is further urged that Claim 6 (and similarly for Claims 13 and 20) has been erroneously rejected due to this additional claimed feature which is not taught or suggested by the cited references.

3. *Claims 7 and 14*

Appellants initially urge error in the rejection of Claim 7 (and similarly for Claim 14) for reasons given above with respect to Claim 1 (of which Claim 7 depends upon), and urge that the additional cited Harrison reference does not overcome the numerous teaching deficiencies identified above with respect to Claim 1.

Further with respect to Claim 7 (and similarly for Claim 14), it is urged that none of the cited references teach or suggest the claimed feature of "wherein the media access control address and the subnet mask are stored together in a record for both a name-to-address file and an address-to-name file". In rejecting Claim 7, the Examiner states that all of the features of Claim 7 are taught by Harrison at paragraph [0191]. Appellants show that there, Harrison states:

> "[0191] DNS: Domain Name System--The on-line distributed database system used to map human-readable machine names into IP addresses. DNS servers throughout the connected Internet implement a hierarchical namespace that allows sites freedom in assigning machine names and addresses. DNS also supports separate mappings between mail destinations and IP addresses."

As can be seen, this cited passage does not teach <u>any</u> *type of storing operation at all.* In addition, this cited passage does not teach the storing of a media access control address. In addition, this cited passage does not teach the storing of a subnet mask. As described above with respect to Claim 6, DNS servers (until the present invention) did not have access to this type of information such as media access control address and subnet mask. Thus, the teachings of a DNS server in this cited passage does not teach or otherwise suggest the storing of either a media access control address or a subnet a mask – and therefore does not teach or otherwise suggest the storing of a media access control address and a subnet mask *together in a record*, as per the features of Claim 7.

Still further, while this cited passage alludes to 'mappings' between mail destinations and

IP addresses, such 'mapping' does not teach or suggest both a name-to-address file and an address-to-name file (**two-way mapping**), or the storing of both a media access control address and a subnet mask *in both of these (missing) files*. Instead, this passage merely describes a **one-way mapping** of human-readable machine names to IP addresses. Thus, it is further urged that Claim 7 (and similarly for Claim 14) has been erroneously rejected, as *none of the numerous features* recited in such claim are taught or suggested by the cited references.

D. GROUND OF REJECTION 4 (Claims 4, 11 and 18)

Claims 4, 11 and 18 stand rejected under 35 U.S.C. § 103 as being unpatentable over Hutchison et al. (U.S. Patent No. 7,249,191 B1), hereinafter "Hutchison" in view of Matsuda et al. (U.S. Patent No. 7,039,688 B2), hereinafter "Matsuda".

1. *Claims 4, 11 and 18*

Appellants initially urge error in the rejection of Claim 4 (and similarly for Claims 11 and 18) for reasons given above with respect to Claim 1 (of which Claim 4 depends upon), and urge that the additional cited Matsuda reference does not overcome the teaching deficiencies identified above with respect to Claim 1.

Further with respect to Claim 4 (and similarly for Claims 11 and 18), it is urged that none of the cited references teach or suggest the claimed feature of "wherein the dynamic host configuration protocol server obtains the media access control address and the subnet mask *from a remote computer when the remote computer requests an address from the dynamic host configuration protocol server*". In rejecting Claim 4, the Examiner states that all of the Claim 4 features are taught by Matsuda at Figure 7, Element 704; Col. 12, lines 46-52; Col. 12, lines 66-67; and Col. 13, lines 1-5. Appellants urge that these cited passages make no mention of any type of *subnet mask*. Rather, a *media access control address* is described. Because Claim 4 recites both a media access control address as well as a subnet mask (and associated operations being performed on both of these two expressly enumerated items), per the features of Claim 4 the 'media access control address' is a different item/thing from the 'subnet mask'. Therefore, it is not proper to interpret Matsuda's teaching of a media access control address to be *both* the claimed media access control address *and* the claimed subnet mask as they are two specific items

expressly enumerated in the claim. Restated, if a media access control address were the same as, or a superset of, a subnet mask, then the claim would only need to recite one and not the other. However, they are not the same, and the claim therefore explicitly recites both of these as being separate elements. Quite simply, a teaching of one (media access control address) does not teach or suggest the other (subnet mask). This can also be seen in Appellants' Figure 5, where a DNS record 500 includes *both* a MAC address 502 *and* a subnet mask 504. While it may be true that certain network operations require both of these separate items in order to function properly, the DHCP server operations described by the cited Matsuda reference have no such requirement of using both (e.g., it is typically a network router that is concerned with the subnet mask, as this subnet mask is used during a routing operation – a DHCP server has no such concern or use for a subnet mask). Thus, Matsuda's teaching of a DHCP server using a media access control address does not teach or suggest the claimed subnet mask features of Claim 4. Thus, it is further shown that Claim 4 has been erroneously rejected due to these additional claimed features that are not taught or suggested by the cited references.

E. GROUND OF REJECTION 5 (Claims 5, 12, 19 and 21)

Claims 5, 12, 19 and 21 stand rejected under 35 U.S.C. § 103 as being unpatentable over Hutchison et al. (U.S. Patent No. 7,249,191 B1), hereinafter "Hutchison" in view of Bahl. (U.S. Patent No. 6,957,276 B1), hereinafter "Bahl".

1. *Claims 5, 12 and 19*

Appellants initially urge error in the rejection of Claim 5 (and similarly for Claims 12 and 19) for reasons given above with respect to Claim 1 (of which Claim 5 depends upon), and urge that the additional cited Bahl reference does not overcome the teaching deficiencies identified above with respect to Claim 1.

Further with respect to Claim 5, it is urged that none of the cited references teach or suggest the claimed feature of "wherein the media access control address and the subnet are received from a user submitting the media access control address and the subnet mask and are stored in a data processing system for the data processing system". As can be seen, a *user* is involved in the operations recited in Claim 5. Such manual intervention by a user may be required in certain situations where a DHCP server is not used to dynamically assign an IP

address to a device, but instead a static IP address is used (Specification page 12, lines 5-24; page 14, line 29 – page 15, line 10). The manual user intervention features recited in Claim 5 accommodate such a scenario. Accordingly, per the features of Claim 5 the ***user*** *submits both a media access control address as well as a subnet mask*, and both the media access control address and the subnet mask are stored in the data processing system. In rejecting Claim 5, the Examiner states that all of the features of Claim 5 are taught by Bahl at Col. 9, lines 1-9. Appellants urge that there, Bahl states:

> "As illustrated in FIG. 2, when a DHCP client machine 200 initially boots onto the network, it transmits a DHCP DISCOVER 202 to the DHCP server 204 in an attempt to obtain an IP address. The DHCP server 204 analyzes the DISCOVER request 202 to determine the type of IP address to be assigned thereto. The DHCP server 204 analyzes the media access control (MAC) address and the client identifier field for the DHCP client 200 that has sent the DISCOVER request 202."

As can be seen, this cited passage describes a *traditional, automatic dynamic IP address assignment* being performed by a DHCP server. In contrast, per the features of Claim 5, a *manual user operation* is claimed, whereby a user submits information that is to be stored in the data processing system. Quite simply, an automated dynamic IP address assignment as described by the teachings of the cited reference does not teach or otherwise suggest any type of manual user operations as provided by the features of Claim 5.

Still further, even assuming arguendo that this cited passage does describe a manual user operation (which Appellants deny), even then there would still be no teaching or suggestion of operations pertaining to *both* a media access control address *as well as* a subnet mask. This cited passage only makes mention of a media access control address and a client identifier field. There is no mention of any type of subnet mask. As described above with respect to Claim 4, and as depicted by Appellants' Figure 5, a media access control address and a subnet mask are separate items – and used differently as one pertains to a *physical* address and the other is used with a *logical* address - and the teaching of one (media access control address) does not teach or suggest

the other (subnet mask). Accordingly, it is further urged that Claim 5 (and similarly for Claims 12 and 19) has been erroneously rejected due to the additional claimed features recited in Claim 5 that are not taught or suggested by the cited references.

2. *Claim 21*

With respect to Claim 21, the Examiner has provided no reason or explanation of how the references read on the features of Claim 21. The Examiner, in rejecting Claim 21, merely relies on the following statement:

> "Hutchinson et al., Bullman et al., Harrison et al. and Matsuda et al. were cited in previous rejections, the teachings that are applicable, hereby incorporated by reference."

Appellants urged that Claim 21 was newly added in the previous response filed by Appellants on January 21, 2008 and thus has not previously been examined by the Examiner. Hence, the Examiner's mere reliance on the reasoning given in the rejection of Claims 5, 12 and 19 fails to address or discuss the specific features pertaining to newly added Claim 21. Thus, Claim 21 has been erroneously rejected as a proper prima facie obviousness has not been established.[2]

In addition, none of the cited references either singularly or in combination teach or suggest the claimed features of "wherein the step of identifying a media access control address and a subnet mask using the request comprises identifying a media access control address and a subnet mask for the remote computer using the request, and wherein the step of returning a response to the requestor comprises returning a response to the requestor, *wherein the response includes the media access control address and the subnet mask for the remote computer*" (emphasis added by Appellants). Thus, it is further shown that Claim 21 has been erroneously rejected.

---

[2] In rejecting claims under 35 U.S.C. Section 103, the examiner bears the initial burden of presenting a prima facie case of obviousness. *In re Oetiker*, 977 F.2d 1443, 1445, 24 USPQ2d 1443, 1444 (Fed. Cir. 1992). Only if that burden is met, does the burden of coming forward with evidence or argument shift to the applicant. *Id.*

F. CONCLUSION

As shown above, the Examiner has failed to state valid rejections against any of the claims. Therefore, Appellants request that the Board of Patent Appeals and Interferences reverse the rejections. Additionally, Appellants request that the Board direct the Examiner to allow the claims.

/Wayne P. Bailey/

Wayne P. Bailey
Reg. No. 34,289
Yee & Associates, P.C.
P.O. Box 802333
Dallas, TX 75380
(972) 385-8777

CLAIMS APPENDIX

The text of the claims involved in the appeal is as follows:

1. A method in a data processing system for providing host information, the method comprising:

receiving a request for host information for a remote computer from a requestor wherein the request includes one of a host name or an Internet Protocol address and is received from the requestor;

identifying a media access control address and a subnet mask using the request; and

returning a response to the requestor, wherein the response includes the media access control address and the subnet mask.

2. The method of claim 1, wherein the requestor generates a wake-up packet using the host information and sends the wake-up packet to the remote computer.

3. The method of claim 1, wherein the media access control address and the subnet mask are received from a dynamic host configuration protocol server and are stored in the data processing system.

4. The method of claim 1, wherein the dynamic host configuration protocol server obtains the media access control address and the subnet mask from a remote computer when the remote computer requests an address from the dynamic host configuration protocol server.

5. The method of claim 1, wherein the media access control address and the subnet are received from a user submitting the media access control address and the subnet mask and are stored in the data processing system.

6. The method of claim 1, wherein the data processing system is a domain name server.

7. The method of claim 1, wherein the media access control address and the subnet mask are stored together in a record for both a name-to-address file and an address-to-name file.

8. A data processing system for providing host information, the data processing system comprising:

receiving means for receiving, by the data processing system, a request for host information for a remote computer from a requestor wherein the request includes one of a host name or an Internet Protocol address and is received from the requestor;

identifying means for identifying, by the data processing system, a media access control address and a subnet mask for the remote computer using the request; and

returning means for returning, by the data processing system, a response to the requestor, wherein the response includes the media access control address and the subnet mask for the remote computer.

9. The data processing system of claim 8, wherein the requestor generates a wake-up packet using the host information and sends the wake-up packet to the remote computer.

10. The data processing system of claim 8, wherein the media access control address and the subnet mask are received from a dynamic host configuration protocol server and are stored in the data processing system.

11. The data processing system of claim 8, wherein the dynamic host configuration protocol server obtains the media access control address and the subnet mask from a remote computer when the remote computer requests an address from the dynamic host configuration protocol server.

12. The data processing system of claim 8, wherein the media access control address and the subnet are received from a user submitting the media access control address and the subnet mask and are stored in the data processing system.

13. The data processing system of claim 8, wherein the data processing system is a domain name server.

14. The data processing system of claim 8, wherein the media access control address and the subnet mask are stored together in a record for both a name-to-address file and an address-to-name file.

15. A computer program product encoded in a computer readable storage medium and operable in a data processing system for providing host information, the computer program product comprising:

first instructions for receiving, by the data processing system, a request for host information for a remote computer from a requestor wherein the request includes one of a host name or an Internet Protocol address and is received from the requestor;

second instructions for identifying, by the data processing system, a media access control address and a subnet mask for the remote computer using the request; and

third instructions for returning, by the data processing system, a response to the requestor, wherein the response includes the media access control address and the subnet mask for the remote computer.

16. The computer program product of claim 15, wherein the requestor generates a wake-up packet using the host information and sends the wake-up packet to the remote computer.

17. The computer program product of claim 15, wherein the media access control address and the subnet mask are received from a dynamic host configuration protocol server and are stored in the data processing system.

18. The computer program product of claim 15, wherein the dynamic host configuration protocol server obtains the media access control address and the subnet mask from a remote computer when the remote computer requests an address from the dynamic host configuration protocol server.

19. The computer program product of claim 15, wherein the media access control address and the subnet are received from a user submitting the media access control address and the subnet mask and are stored in the data processing system.

20. The computer program product of claim 15, wherein the data processing system is a domain name server.

21. The method of Claim 1, wherein the step of identifying a media access control address and a subnet mask using the request comprises identifying a media access control address and a subnet mask for the remote computer using the request, and wherein the step of returning a response to the requestor comprises returning a response to the requestor, wherein the response includes the media access control address and the subnet mask for the remote computer.

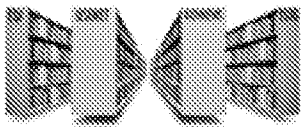
EVIDENCE APPENDIX

This appeal brief presents ATTACHMENTS A, B, C and D

RELATED PROCEEDINGS APPENDIX

This appeal has no related proceedings.

ATTACHMENT A



RFC 917 (RFC917)

Internet RFC/STD/FYI/BCP Archives

[RFC Index | RFC Search | Usenet FAQs | Web FAQs | Documents | Cities]

Alternate Formats: rfc917.txt | rfc917.txt.pdf

Comment on RFC 917

RFC 917 - Internet subnets

Network Working Group
Request for Comments: 917

Jeffrey Mogul
Computer Science Department
Stanford University
October 1984

INTERNET SUBNETS

Status Of This Memo

This RFC suggests a proposed protocol for the ARPA-Internet community, and requests discussion and suggestions for improvements. Distribution of this memo is unlimited.

Overview

We discuss the utility of "subnets" of Internet networks, which are logically visible sub-sections of a single Internet network. For administrative or technical reasons, many organizations have chosen to divide one Internet network into several subnets, instead of acquiring a set of Internet network numbers.

We propose procedures for the use of subnets, and discuss approaches to solving the problems that arise, particularly that of routing.

Acknowledgment

This proposal is the result of discussion with several other people. J. Noel Chiappa, Chris Kent, and Tim Mann, in particular, provided important suggestions.

1. Introduction

The original view of the Internet universe was a two-level hierarchy: the top level the catenet as a whole, and the level below it a collection of "Internet Networks", each with its own Network Number. (We do not mean that the Internet has a hierarchical topology, but

that the interpretation of addresses is hierarchical.)

While this view has proved simple and powerful, a number of organizations have found it inadequate and have added a third level to the interpretation of Internet addresses. In this view, a given Internet Network might (or might not) be divided into a collection of subnets.

The original, two-level, view carries a strong presumption that, to a host on an Internet network, that network may be viewed as a single edge; to put it another way, the network may be treated as a "black box" to which a set of hosts is connected. This is true of the

ARPANET, because the IMPs mask the use of specific links in that network. It is also true of most local area network (LAN) technologies, such as Ethernet or ring networks.

However, this presumption fails in many practical cases, because in moderately large organizations (e.g., Universities or companies with more than one building) it is often necessary to use more than one LAN cable to cover a "local area". For example, at this writing there are eighteen such cables in use at Stanford University, with more planned.

There are several reasons why an organization might use more than one cable to cover a campus:

- Different technologies: Especially in a research environment, there may be more than one kind of LAN in use; e.g., an organization may have some equipment that supports Ethernet, and some that supports a ring network.

- Limits of technologies: Most LAN technologies impose limits, based electrical parameters, on the number of hosts connected, and on the total length of the cable. It is easy to exceed these limits, especially those on cable length.

- Network congestion: It is possible for a small subset of the hosts on a LAN to monopolize most of the bandwidth. A common solution to this problem is to divide the hosts into cliques of high mutual communication, and put these cliques on separate cables.

- Point-to-Point links: Sometimes a "local area", such as a university campus, is split into two locations too far apart to connect using the preferred LAN technology. In this case, high-speed point-to-point links might connect several LANs.

An organization that has been forced to use more than one LAN has three choices for assigning Internet addresses:

1. Acquire a distinct Internet network number for each cable.

2. Use a single network number for the entire organization, but assign host numbers without regard to which LAN a host is on. (We will call this choice "transparent subnets".)

3. Use a single network number, and partition the host address space by assigning subnet numbers to the LANs. ("Explicit subnets".)

Each of these approaches has disadvantages. The first, although not requiring any new or modified protocols, does result in an explosion in the size of Internet routing tables. Information about the internal details of local connectivity is propagated everywhere, although it is of little or no use outside the local organization. Especially as some current gateway implementations do not have much space for routing tables, it would be nice to avoid this problem.

The second approach requires some convention or protocol that makes the collection of LANs appear to be a single Internet network. For example, this can be done on LANs where each Internet address is translated to a hardware address using an Address Resolution Protocol (ARP), by having the bridges between the LANs intercept ARP requests for non-local targets. However, it is not possible to do this for all LAN technologies, especially those where ARP protocols are not currently used, or if the LAN does not support broadcasts. A more fundamental problem is that bridges must discover which LAN a host is on, perhaps by using a broadcast algorithm. As the number of LANs grows, the cost of broadcasting grows as well; also, the size of translation caches required in the bridges grows with the total number of hosts in the network.

The third approach addresses the key problem: existing standards assume that all hosts on an Internet local network are on a single cable. The solution is to explicitly support subnets. This does have a disadvantage, in that it is a modification of the Internet Protocol, and thus requires changes to IP implementations already in use (if these implementations are to be used on a subnetted network.) However, we believe that these changes are relatively minor, and once made, yield a simple and efficient solution to the problem. Also, the approach we take in this document is to avoid any changes that would be incompatible with existing hosts on non-subnetted networks.

Further, when appropriate design choices are made, it is possible for hosts which believe they are on a non-subnetted network to be used on a subnetted one, as will be explained later. This is useful when it is not possible to modify some of the hosts to support subnets explicitly, or when a gradual transition is preferred. Because of this, there seems little reason to use the second approach listed above.

The rest of this document describes approaches to subnets of Internet Networks.

1.1. Terminology

To avoid either ambiguity or prolixity, we will define a few terms, which will be used in the following sections:

Catenet

The collection of connected Internet Networks

Network

A single Internet network (that may or may not be divided into subnets.)

Subnet

A subnet of an Internet network.

Network Number

As in [8].

Local Address

The bits in an Internet address not used for the network number; also known as "rest field".

Subnet Number

A number identifying a subnet within a network.

Subnet Field

The bit field in an Internet address used for the subnet number.

Host Field

The bit field in an Internet address used for denoting a specific host.

Gateway

A node connected to two or more administratively distinct networks and/or subnets, to which hosts send datagrams to be forwarded.

Bridge

A node connected to two or more administratively indistinguishable but physically distinct subnets, that automatically forwards datagrams when necessary, but whose existence is not know to other hosts. Also called a "software repeater".

2. Standards for Subnet Addressing

Following the division presented in [2], we observe that subnets are fundamentally an issue of addressing. In this section, we first describe a proposal for interpretation of Internet Addressing to support subnets. We then discuss the interaction between this address format and broadcasting; finally, we present a protocol for discovering what address interpretation is in use on a given network.

2.1. Interpretation of Internet Addresses

Suppose that an organization has been assigned an Internet network number, has further divided that network into a set of subnets, and wants to assign host addresses: how should this be done? Since there are minimal restrictions on the assignment of the "local address" part of the Internet address, several approaches have been proposed for representing the subnet number:

1. Variable-width field: Any number of the bits of the local address part are used for the subnet number; the size of this field, although constant for a given network, varies from network to network. If the field width is zero, then subnets are not in use.

2. Fixed-width field: A specific number of bits (e.g., eight) is used for the subnet number, if subnets are in use.

3. Self-encoding variable-width field: Just as the width (i.e., class) of the network number field is encoded by its high-order bits, the width of the subnet field is similarly encoded.

4. Self-encoding fixed-width field: A specific number of bits is is used for the subnet number. Subnets are in use if the high-order bit of this field is one; otherwise, the entire local address part is used for host number.

Since there seems to be no advantage in doing otherwise, all these schemes place the subnet field as the most significant field in

the local address part. Also, since the local address part of a Class C address is so small, there is little reason to support subnets of other than Class A and Class B networks.

What criteria can we use to choose one of these four schemes? First, do we want to use a self-encoding scheme; that is, should it be possible to tell from examining an Internet address if it refers to a subnetted network, without reference to any other information?

One advantage to self-encoding is that it allows one to determine if a non-local network has been divided into subnets. It is not clear that this would be of any use. The principle advantage, however, is that no additional information is needed for an implementation to determine if two addresses are on the same subnet. However, this can also be viewed as a disadvantage: it may cause problems for non-subnetted networks which have existing host numbers that use arbitrary bits in the local address part <1>. In other words, it is useful to be able control whether a network is subnetted independently from the assignment of host addresses. Another disadvantage of any self-encoding scheme is that it reduces the local address space by at least a factor of two.

If a self-encoding scheme is not used, it is clear that a

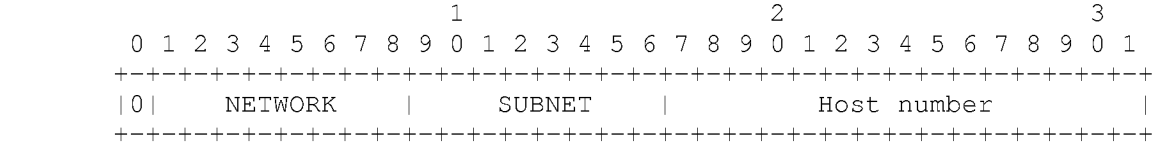
variable-width subnet field is appropriate. Since there must in any case be some per-network "flag" to indicate if subnets are in use, the additional cost of using an integer (the subnet field width) instead of a boolean is negligible. The advantage of using a variable-width subnet field is that it allows each organization to choose the best way to allocate relatively scarce bits of local address to subnet and host numbers.

Our proposal, therefore, is that the Internet address be interpreted as:

<network-number><subnet-number><host-number>

where the <network-number> field is as in [8], the <host-number> field is at least one bit wide, and the width of the <subnet-number> field is constant for a given network. No further structure is required for the <subnet-number> or <host-number> fields. If the width of the <subnet-number> field is zero, then the network is not subnetted (i.e., the interpretation of [8] is used.)

For example, on a Class A network with an eight bit wide subnet field, an address is broken down like this:



We expect that, for reasons of simplicity and efficient implementation, that most organizations will choose a subnet field width that is a multiple of eight bits. However, an implementation must be prepared to handle other possible widths.

We reject the use of "recursive subnets", the division of the host field into "sub-subnet" and host parts, because:

- There is no obvious need for a four-level hierarchy.
- The number of bits available in an IP address is not large enough to make this useful in general.
- The extra mechanism required is complex.

2.2. Changes to Host Software to Support Subnets

In most implementations of IP, there is code in the module that handles outgoing packet that does something like:

```
IF ip_net_number(packet.ip_dest) = ip_net_number(my_ip_addr)
    THEN
        send_packet_locally(packet, packet.ip_dest)
    ELSE
        send_packet_locally(packet,
           gateway_to(ip_net_number(packet.ip_dest)))
```

(If the code supports multiple connected networks, it will be more complicated, but this is irrelevant to the current discussion.)

To support subnets, it is necessary to store one more 32-bit quantity, called my_ip_mask. This is a bit-mask with bits set in the fields corresponding to the IP network number, and additional bits set corresponding to the subnet number field. For example, on a Class A network using an eight-bit wide subnet field, the mask would be 255.255.0.0.

The code then becomes:


```
IF bitwise_and(packet.ip_dest, my_ip_mask)
                = bitwise_and(my_ip_addr, my_ip_mask)
    THEN
        send_packet_locally(packet, packet.ip_dest)
    ELSE
        send_packet_locally(packet,
           gateway_to(bitwise_and(packet.ip_dest, my_ip_mask)))
```

Of course, part of the expression in the conditionally can be pre-computed.

It may or may not be necessary to modify the "gateway_to" function, so that it performs comparisons in the same way.

To support multiply-connected hosts, the code can be changed to keep the "my_ip_addr" and "my_ip_mask" quantities on a per-interface basis; the expression in the conditional must then be evaluated for each interface.

2.3. Subnets and Broadcasting

In the absence of subnets, there are only two kinds of broadcast possible within the Internet Protocol <2>: broadcast to all hosts on a specific network, or broadcast to all hosts on "this network"; the latter is useful when a host does not know what network it is on.

When subnets are used, the situation becomes slightly more complicated. First, the possibility now exists of broadcasting to a specific subnet. Second, broadcasting to all the hosts on a subnetted network requires additional mechanism; in [6] the use of "Reverse Path Forwarding" [3] is proposed. Finally, the interpretation of a broadcast to "this network" is that it should not be forwarded outside of the original subnet.

Implementations must therefore recognize three kinds of broadcast addresses, in addition to their own host addresses:

This physical network

A destination address of all ones (255.255.255.255) causes the a datagram to be sent as a broadcast on the local physical network; it must not be forwarded by any gateway.

Specific network

The destination address contains a valid network number; the local address part is all ones (e.g., 36.255.255.255).

Specific subnet

The destination address contains a valid network number and a valid subnet number; the host field is all ones (e.g., 36.40.255.255).

For further discussion of Internet broadcasting, see [6].

One factor that may aid in deciding whether to use subnets is that it is possible to broadcast to all hosts of a subnetted network with a single operation at the originating host. It is not possible to broadcast, in one step, to the same set of hosts if they are on distinct networks.

2.4. Determining the Width of the Subnet Field

How can a host (or gateway) determine what subnet field width is in use on a network to which it is connected? The problem is analogous to several other "bootstrapping" problems for Internet hosts: how a host determines its own address, and how it locates a gateway on its local network. In all three cases, there are two basic solutions: "hardwired" information, and broadcast-based protocols.

"Hardwired" information is that available to a host in isolation from a network. It may be compiled-in, or (preferably) stored in a disk file. However, for the increasingly common case of a diskless workstation that is bootloaded over a LAN, neither hard-wired solution is satisfactory. Instead, since most LAN technology supports broadcasting, a better method is for the newly-booted host to broadcast a request for the necessary information. For example, for the purpose of determining its Internet address, a host may use the "Reverse Address Resolution Protocol" [4].

We propose to extend the ICMP protocol [9] by adding a new pair of ICMP message types, "Address Format Request" and "Address Format Reply", analogous to the "Information Request" and "Information Reply" ICMP messages. These are described in detail in Appendix I.

The intended use of these new ICMPs is that a host, when booting,

broadcast an "Address Format Request" message <3>. A gateway (or a host acting in lieu of a gateway) that receives this message responds with an "Address Format Reply". If there is no indication in the request which host sent it (i.e., the IP Source Address is zero), the reply is broadcast as well. The requesting host will hear the response, and from it determine the width of the subnet field.

Since there is only one possible value that can be sent in an "Address Format Reply" on any given LAN, there is no need for the requesting host to match the responses it hears against the request it sent; similarly, there is no problem if more than one gateway responds. We assume that hosts reboot infrequently, so the broadcast load on a network from use of this protocol should be small.

If a host is connected to more than one LAN, it must use this protocol on each, unless it can determine (from a response on one of the LANs) that several of the LANs are part of the same network, and thus must have the same subnet field width.

One potential problem is what a host should do if it receives no response to its "Address Format Request", even after a reasonable number of tries. Three interpretations can be placed on the situation:

1. The local net exists in (permanent) isolation from all other nets.

2. Subnets are not in use, and no host supports this ICMP request.

3. All gateways on the local net are (temporarily) down.

The first and second situations imply that the subnet field width is zero. In the third situation, there is no way to determine what the proper value is; the safest choice is thus zero. Although this might later turn out to be wrong, it will not prevent transmissions that would otherwise succeed. It is possible for a host to recover from a wrong choice: when a gateway comes up, it should broadcast an "Address Format Reply"; when a host receives such a message that disagrees with its guess, it should adjust its data structures to conform to the received value. No host or gateway should send an "Address Format Reply" based on a "guessed" value.

Finally, note that no host is required to use this ICMP protocol to discover the subnet field width; it is perfectly reasonable for a host with non-volatile storage to use stored information.

3. Subnet Routing Methods

One problem that faces all Internet hosts is how to determine a route to another host. In the presence of subnets, this problem is only slightly modified.

The use of subnets means that there are two levels to the routing process, instead of one. If the destination host is on the same network as the source host, the routing decision involves only the subnet gateways between the hosts. If the destination is on a different network, then the routing decision requires the choice both of a gateway out of the source host's network, and of a route within the network to that gateway.

Fortunately, many hosts can ignore this distinction (and, in fact, ignore all routing choices) by using a "default" gateway as the initial route to all destinations, and relying on ICMP Host Redirect messages to define more appropriate routes. However, this is not an efficient method for a gateway or for a multi-homed host, since a redirect may not make up for a poor initial choice of route. Such hosts should use a routing information exchange protocol, but that is beyond the scope of this document; in any case, the problem arises even when subnets are not used.

The problem for a singly-connected host is thus to find at least one neighbor gateway. Again, there are basic two solutions to this: use hard-wired information, or use broadcasts. We believe that the neighbor-gateway acquisition problem is the same with or without subnets, and thus the choice of solution is not affected by the use of subnets.

However, one problem remains: a source host must determine if datagram to a given destination address must be sent via a gateway, or sent directly to the destination host. In other words, is the destination host on the same physical network as the source? This particular phase of the routing process is the only one that requires an implementation to be explicitly aware of subnets; in fact, if broadcasts are not used, it is the only place where an Internet implementation must be modified to support subnets.

Because of this, it is possible to use some existing implementations without modification in the presence of subnets <4>. For this to work, such implementations must:

- Be used only on singly-homed hosts, and not as a gateway.

- Be used on a broadcast LAN.

- Use an Address Resolution Protocol (ARP), such [7].

- Not be required to maintain connections in the case of gateway crashes.

In this case, one can modify the ARP server module in a subnet gateway so that when it receives an ARP request, it checks the target Internet address to see if it is along the best route to the target. If it is, it sends to the requesting host an ARP response indicating its own hardware address. The requesting host thus believes that it knows the hardware address of the destination host, and sends packets to that address. In fact, the packets are received by the gateway, and forwarded to the destination host by the usual means.

This method requires some blurring of the layers in the gateways, since the ARP server and the Internet routing table would normally not have any contact. In this respect, it is somewhat unsatisfactory. Still, it is fairly easy to implement, and does not have significant performance costs. One problem is that if the original gateway crashes, there is no way for the source host to choose an alternate route even if one exists; thus, a connection that might otherwise have been maintained will be broken.

One should not confuse this method of "ARP-based subnetting" with the superficially similar use of ARP-based bridges. ARP-based subnetting is based on the ability of a gateway to examine an IP address and deduce a route to the destination, based on explicit subnet topology. In other words, a small part of the routing decision has been moved from the source host into the gateway. An ARP-based bridge, in contrast, must somehow locate each host without any assistance from a mapping between host address and topology. Systems built out of ARP-based bridges should not be referred to as "subnetted".

N.B.: the use of ARP-based subnetting is complicated by the use of broadcasts. An ARP server [7] should never respond to a request whose target is a broadcast address. Such a request can only come from a host that does not recognize the broadcast address as such, and so honoring it would almost certainly lead to a forwarding loop. If there are N such hosts on the physical network that do not recognize this address as a broadcast, then a packet sent with a Time-To-Live of T could potentially give rise to T**N spurious re-broadcasts.

4. Case Studies

In this section, we briefly sketch how subnets have been used by several organizations.

4.1. Stanford University

At Stanford, subnets were introduced initially for historical reasons. Stanford had been using the Pup protocols [1] on a collection of several Experimental Ethernets [5] since 1979, several years before Internet protocols came into use. There were a number of Pup gateways in service, and all hosts and gateways acquired and exchanged routing table information using a simple broadcast protocol.

When the Internet Protocol was introduced, the decision was made to use an eight-bit wide subnet number; Internet subnet numbers were chosen to match the Pup network number of a given Ethernet, and the Pup host numbers (also eight bits) were used as the host field of the Internet address.

The Pup-only gateways were then modified to forward Internet datagrams according to their Pup routing tables; they otherwise had no understanding of Internet packets and in fact did not adjust the Time-to-live field in the Internet header. This seems to be acceptable, since bugs that caused forwarding loops have not appeared. The Internet hosts that are multi-homed and thus can serve as gateways do adjust the Time-to-live field; since all of the currently also serve as Pup gateways, no additional routing information exchange protocol was needed.

Internet host implementations were modified to understand subnets (in several different ways, but with identical effects). Since all already had Pup implementations, the Internet routing tables were maintained by the same process that maintained the Pup routing tables, simply translating the Pup network numbers into Internet subnet numbers.

When 10Mbit Ethernets were added, the gateways were modified to use the ARP-based scheme described in an earlier section; this allowed unmodified hosts to be used on the 10Mbit Ethernets.

IP subnets have been in use since early 1982; currently, there are about 330 hosts, 18 subnets, and a similar number of subnet gateways in service. Once the Pup-only gateways are converted to be true Internet gateways, an Internet-based routing exchange protocol will be introduced, and Pup will be phased out.

4.2. MIT

MIT was the first IP site to accumulate a large collection of local network links. Since this happened before network numbers were divided into classes, to have assigned each link at MIT its own IP network number would have used up a good portion of the available address space. MIT decided to use one IP network number, and to manage the 24-bit "rest" field itself, by dividing it into three 8-bit fields; "subnet", "reserved, must be zero", and "host". Since the CHAOS protocol already in use at MIT used an 8-bit subnet number field, it was possible to assign each link the same subnet number in both protocols. The IP host field was set to 8 bits since most available local net hardware at that point used 8 bit addresses, as did the CHAOS protocol; it was felt that reserving some bits for the future was wise.

The initial plan was to use a dynamic routing protocol between the IP subnet gateways; several such protocols have been mooted but nobody has bothered to implement one; static routing tables are still used. It is likely that this change will finally be made soon.

To solve the problem that imported IP software always needed modification to work in the subnetted environment, MIT searched for a model of operation that led to the least change in host IP software. This led to a model where IP gateways send ICMP Host Redirects rather than Network Redirects. All internal MIT IP gateways now do so. With hosts that can maintain IP routing tables for non-local communication on a per host basis, this hides most of the subnet structure. The "minimum adjustment" for host software to work correctly in both subnetted and non-subnetted environments is the bit-mask algorithm mentioned earlier.

MIT has no immediate plans to move toward a single "approved" protocol; this is due partly to the degree of local autonomy and the amount of installed software, and partly to the lack of a single prominent industry standard. Rather, the approach taken has been to provide a single set of physical links and packet switches, and to layer several "virtual" protocol nets atop the single set of links. MIT has had some bad experiences with trying to exchange routing information between protocols and wrap one protocol in another; the general approach is to keep the protocols strictly separated except for sharing the basic hardware. Using ARP to hide the subnet structure is not much in favor; it is felt that this overloads the address resolution operation. In a complicated system (i.e. one with loops, and variant link speeds),

a more sophisticated information interchange will be needed between gateways; making this an explicit mechanism (but one insulated from the hosts) was felt to be best.

4.3. Carnegie-Mellon University

CMU uses a Class B network currently divided into 11 physical subnets (two 3Mbit Experimental Ethernets, seven 10Mbit Ethernets, and two ProNet rings.) Although host numbers are assigned so that all addresses with a given third octet will be on the same subnet (but not necessarily vice versa), this is essentially an administrative convenience. No software currently knows the specifics of this allocation mechanism or depends on it to route between cables.

Instead, an ARP-based bridge scheme is used. When a host broadcasts an ARP request, all bridges which receive it cache the original protocol address mapping and then forward the request (after the appropriate adjustments) as an ARP broadcast request onto each of their other connected cables. When a bridge receives a non-broadcast ARP reply with a target protocol address not its own, it consults its ARP cache to determine the cable onto which the reply should be forwarded. The bridges thus attempt to transparently extend the ARP protocol into a heterogenous multi-cable environment. They are therefore required to turn ARP broadcasts on a single cable into ARP broadcasts on all other connected cables even when they "know better". This algorithm works only in the absence of cycles in the network connectivity graph (which is currently the case). Work is underway to replace this simple-minded algorithm with a protocol implemented among the bridges, in support of redundant paths and to reduce the collective broadcast load. The intent is to retain the ARP base and host transparency, if possible.

Implementations supporting the 3Mbit Ethernet and 10Mb proNET ring at CMU use RFC-826 ARP (instead of some wired-in mapping such as simply using the 8-bit hardware address as the the fourth octet of the IP address).

Since there are currently no redundant paths between cables, the issue of maintaining connections across bridge crashes is moot. With about 150 IP-capable hosts on the net, the bridge caches are still of reasonable size, and little bandwidth is devoted to ARP broadcast forwarding.

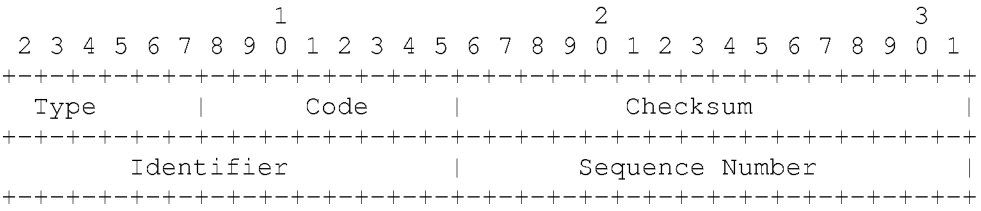
CMU's network is likely to grow from its relatively small, singly-connected configuration centered within their CS/RI

facility to a campus-wide intra-departmental configuration with 5000-10000 hosts and redundant connections between cables. It is possible that the ARP-based bridge scheme will not scale to this size, and a system of explicit subnets may be required. The medium-term goal, however, is an environment into which unmodified

extant (especially 10Mb ethernet based) IP implementations can be imported; the intent is to stay with a host-transparent (thus ARP-based) routing mechanism as long as possible. CMU is concerned that even if subnets become part of the IP standard they will not be widely implemented; this is the major obstacle to their use at CMU.

I. Address Format ICMP

Address Format Request or Address Format Reply



IP Fields:

Addresses

The address of the source in an address format request message will be the destination of the address format reply message. To form an address format reply message, the source address of the request becomes the destination address of the reply, the source address of the reply is set to the replier's address, the type code changed to A2, the subnet field width inserted into the Code field, and the checksum recomputed. However, if the source address in the request message is zero, then the destination address for the reply message should denote a broadcast.

ICMP Fields:

Type

A1 for address format request message

A2 for address format reply message

Code

0 for address format request message

Width of subnet field, in bits, for address format reply message

Checksum

The checksum is the 16-bit one's complement of the one's

complement sum of the ICMP message starting with the ICMP

Type. For computing the checksum, the checksum field should be zero. This checksum may be replaced in the future.

Identifier

An identifier to aid in matching request and replies, may be zero.

Sequence Number

A sequence number to aid in matching request and replies, may be zero.

Description

A gateway receiving an address format request should return it with the Code field set to the number of bits of Subnet number in IP addresses for the network to which the datagram was addressed. If the request was broadcast, the destination network is "this network". The Subnet field width may be from 0 to (31 - N), where N is the width in bits of the IP net number field (i.e., 8, 16, or 24).

If the requesting host does not know its own IP address, it may leave the source field zero; the reply should then be broadcast. Since there is only one possible address format for a network, there is no need to match requests with replies. However, this approach should be avoided if at all possible, since it increases the superfluous broadcast load on the network.

Type A1 may be received from a gateway or a host.

Type A2 may be received from a gateway, or a host acting in lieu of a gateway.

II. Examples

For these examples, we assume that the requesting host has address 36.40.0.123, that there is a gateway at 36.40.0.62, and that on network 36.0.0.0, an 8-bit wide subnet field is in use.

First, suppose that broadcasting is allowed, and that 36.40.0.123 knows its own address. It sends the following datagram:

Source address:	36.40.0.123
Destination address:	36.255.255.255
Protocol:	ICMP = 1
Type:	Address Format Request = A1
Code:	0

36.40.0.62 will hear the datagram, and should respond with this datagram:

Source address:	36.40.0.62
Destination address:	36.40.0.123
Protocol:	ICMP = 1

```
   Type:                    Address Format Reply = A2
   Code:                    8
```

For the following examples, assume that address 255.255.255.255 denotes "broadcast to this physical network", as described in [6].

The previous example is inefficient, because it potentially broadcasts the request on many subnets. The most efficient method, and the one we recommend, is for a host to first discover its own address (perhaps using the "Reverse ARP" protocol described in [4]), and then to send the ICMP request to 255.255.255.255:

```
   Source address:          36.40.0.123
   Destination address:     255.255.255.255
   Protocol:                ICMP = 1
   Type:                    Address Format Request = A1
   Code:                    0
```

The gateway can then respond directly to the requesting host.

Suppose that 36.40.0.123 is a diskless workstation, and does not know even its own host number. It could send the following datagram:


```
   Source address:          0.0.0.0
   Destination address:     255.255.255.255
   Protocol:                ICMP = 1
   Type:                    Address Format Request = A1
   Code:                    0
```

36.40.0.62 will hear the datagram, and should respond with this datagram:

```
   Source address:          36.40.0.62
   Destination address:     36.40.255.255
   Protocol:                ICMP = 1
   Type:                    Address Format Reply = A2
   Code:                    8
```

Note that the gateway uses the narrowest possible broadcast to reply (i.e., sending the reply to 36.255.255.255 would mean that it is transmitted on many subnets, not just the one on which it is needed.) Even so, the overuse of broadcasts presents an unnecessary load to all hosts on the subnet, and so we recommend that use of the "anonymous" (0.0.0.0) source address be kept to a minimum.

If broadcasting is not allowed, we assume that hosts have wired-in information about neighbor gateways; thus, 36.40.0.123 might send this datagram:

```
   Source address:          36.40.0.123
   Destination address:     36.40.0.62
   Protocol:                ICMP = 1
   Type:                    Address Format Request = A1
   Code:                    0
```

36.40.0.62 should respond exactly as in the previous case.

Notes

<1> For example, some host have addresses assigned by concatenating their Class A network number with the low-order 24 bits of a 48-bit Ethernet hardware address.

<2> Our discussion of Internet broadcasting is based on [6].

<3> If broadcasting is not supported, them presumably a host "knows" the address of a neighbor gateway, and should send the ICMP to that gateway.

<4> This is what was referred to earlier as the coexistence of transparent and explicit subnets on a single network.

References

1. D.R. Boggs, J.F. Shoch, E.A. Taft, and R.M. Metcalfe. "Pup: An Internetwork Architecture." IEEE Transactions on Communications COM-28, 4, pp612-624, April 1980.

2. David D. Clark. Names, Addresses, Ports, and Routes. RFC-814, MIT-LCS, July 1982.

3. Yogan K. Dalal and Robert M. Metcalfe. "Reverse Path Forwarding of Broadcast Packets." Comm. ACM 21, 12, pp1040-1048, December 1978.

4. Ross Finlayson, Timothy Mann, Jeffrey Mogul, Marvin Theimer. A Reverse Address Resolution Protocol. RFC-903, Stanford University, June 1984.

5. R.M. Metcalfe and D.R. Boggs. "Ethernet: Distributed Packet Switching for Local Computer Networks." Comm. ACM 19, 7, pp395-404, July 1976. Also CSL-75-7, Xerox Palo Alto Research Center, reprinted in CSL-80-2.

6. Jeffrey Mogul. Broadcasting Internet Datagrams. RFC-919, Stanford University, October 1984.

7. David Plummer. An Ethernet Address Resolution Protocol. RFC-826, Symbolics, September 1982.

8. Jon Postel. Internet Protocol. RFC-791, USC-ISI, September 1981.

9. Jon Postel. Internet Control Message Protocol. RFC-792, USC-ISI, September 1981.

Comment on RFC 917

Previous: RFC 0916 - Reliable Asynchronous Transfer Protocol (RATP)

Next: RFC 0918 - Post Office Protocol

[RFC Index | RFC Search | Usenet FAQs | Web FAQs | Documents | Cities]

ATTACHMENT B

Subnetting

Previous  Next

Subnetting an IP Network can be done for a variety of reasons, including organization, use of different physical media (such as Ethernet, FDDI, WAN, etc.), preservation of address space, and security. The most common reason is to control network traffic. In an Ethernet network, all nodes on a segment see all the packets transmitted by all the other nodes on that segment. Performance can be adversely affected under heavy traffic loads, due to collisions and the resulting retransmissions. A router is used to connect IP networks to minimize the amount of traffic each segment must receive.

Subnet Masking

Applying a subnet mask to an IP address allows you to identify the network and node parts of the address. The network bits are represented by the 1s in the mask, and the node bits are represented by the 0s. Performing a bitwise logical AND operation between the IP address and the subnet mask results in the *Network Address* or Number.
For example, using our test IP address and the default Class B subnet mask, we get:

```
10001100.10110011.11110000.11001000      140.179.240.200   Class B IP Address
11111111.11111111.00000000.00000000      255.255.000.000   Default Class B Subnet Mask
---------------------------------------------------------
10001100.10110011.00000000.00000000      140.179.000.000   Network Address
```

Default subnet masks:

- **Class A** - 255.0.0.0 - 11111111.00000000.00000000.00000000
- **Class B** - 255.255.0.0 - 11111111.11111111.00000000.00000000
- **Class C** - 255.255.255.0 - 11111111.11111111.11111111.00000000

Previous  Next

Updated January 29, 2007

ATTACHMENT C

TCP / IP Suite

IP

RFC 791 http://www.cis.ohio-state.edu/htbin/rfc/rfc791.html

The Internet Protocol (IP), defined by IETF RFC791, is the routing layer datagram service of the TCP/IP suite. All other protocols within the TCP/IP suite, except ARP and RARP, use IP to route frames from host to host. The IP frame header contains routing information and control information associated with datagram delivery.

The IP header structure is as follows:

▲TOP

4	8	16	32 bits
Ver.	IHL	Type of service	Total length
Identification		Flags	Fragment offset
Time to live	Protocol	Header checksum	
Source address			
Destination address			
Option + Padding			
Data			

IP header structure

Version

Version field indicates the format of the Internet header.

IHL

Internet header length is the length of the Internet header in 32-bit words. Points to the beginning of the data. The minimum value for a correct header is 5.

Type of service

Indicates the quality of service desired. Networks may offer service precedence, meaning that they accept traffic only above a certain precedence at times of high load. There is a three-way trade-off between low delay, high reliability and high throughput.

Bits 0-2: Precedence
111 Network control.
110 Internetwork control.
101 CRITIC/ECP.
100 Flash override.
011 Flash.
010 Immediate.
001 Priority.
000 Routine.

Bit 3: Delay

0 Normal delay.
1 Low delay.

Bit 4: Throughput

0 Normal throughput.
1 High throughput.

Bit 5: Reliability

0 Normal reliability.
1 High reliability.

Bits 6-7: Reserved for future use.

Total length
Length of the datagram measured in bytes, including the Internet header and data. This field allows the length of a datagram to be up to 65,535 bytes, although such long datagrams are impractical for most hosts and networks. All hosts must be prepared to accept datagrams of up to 576 bytes, regardless of whether they arrive whole or in fragments. It is recommended that hosts send datagrams larger than 576 bytes only if the destination is prepared to accept the larger datagrams.

Identification
Identifying value assigned by the sender to aid in assembling the fragments of a datagram.

Flags
3 bits. Control flags:

Bit 0 is reserved and must be zero

Bit 1: Don't fragment bit:

0 May fragment.
1 Don't fragment.

Bit 2: More fragments bit:

0 Last fragment.
1 More fragments.

Fragment offset
13 bits. Indicates where this fragment belongs in the datagram. The fragment offset is measured in units of 8 bytes (64 bits). The first fragment has offset zero.

Time to live
Indicates the maximum time the datagram is allowed to remain in the Internet system. If this field contains the value zero, the datagram must be destroyed. This field is modified in Internet header processing. The time is measured in units of seconds. However, since every module that processes a datagram must decrease the TTL by at least one (even if it processes the datagram in less than 1 second), the TTL must be thought of only as an upper limit on the time a datagram may exist. The intention is to cause undeliverable datagrams to be discarded and to bound the maximum datagram lifetime.

Protocol
Indicates the next level protocol used in the data portion of the Internet datagram.

Header checksum
A checksum on the header only. Since some header fields change, e.g., Time To Live, this is recomputed and verified at each point that the Internet header is processed.

Source address / destination address
32 bits each. A distinction is made between names, addresses and routes. A *name* indicates an object to be sought. An *address* indicates the location of the object. A *route* indicates how to arrive at the object. The Internet protocol deals primarily with addresses. It is the task of higher level protocols (such as host-to-host or application) to make the mapping from names to addresses. The Internet module maps Internet addresses to local net addresses. It is the task of lower level procedures (such as local net or gateways) to make the mapping from local net addresses to routes.

Options
Options may or may not appear in datagrams. They must be implemented by all IP modules (host and gateways). What is optional is their transmission in any particular datagram, not their implementation. In some environments, the security option may be

required in all datagrams.

The option field is variable in length. There may be zero or more options. There are two possible formats for an option:

- A single octet of option type.
- An option type octet, an option length octet and the actual option data octets.

The length octet includes the option type octet and the actual option data octets.

The option type octet has 3 fields:

1 bit: Copied flag. Indicates that this option is copied into all fragments during fragmentation:

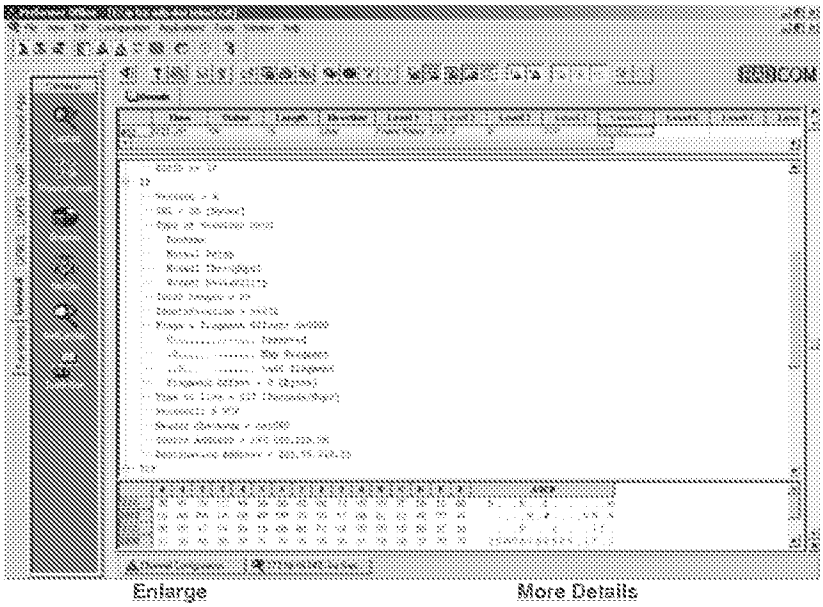
0 Copied.
1 Not copied.

2 bits: Option class

0 Control.
1 Reserved for future use.
2 Debugging and measurement.
3 Reserved for future use.

5 bits: Option number.

Data
IP data or higher layer protocol header.



Enlarge More Details

Interested in more details about testing this protocol? Click here

IPv6

RFC1883 http://www.cis.ohio-state.edu/htbin/rfc/rfc1883.html
RFC1827 http://www.cis.ohio-state.edu/htbin/rfc/rfc1827.html

IP version 6 (IPv6) is a new version of the Internet Protocol based on IPv4. IPv4 and IPv6 are demultiplexed at the media layer. For example, IPv6 packets are carried over Ethernet with the content type 86DD (hexadecimal) instead of IPv4's 0800.

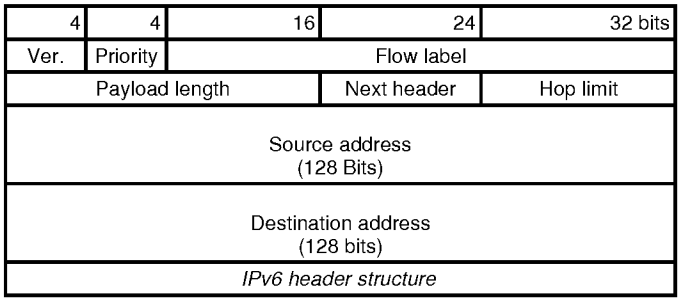
IPv6 increases the IP address size from 32 bits to 128 bits, to support more levels of

addressing hierarchy, a much greater number of addressable nodes and simpler auto-configuration of addresses. Scalability of multicast addresses is introduced. A new type of address called an *anycast address* is also defined, to send a packet to any one of a group of nodes.

Improved support for extensions and options - IPv6 options are placed in separate headers that are located between the IPv6 header and the transport layer header. Changes in the way IP header options are encoded allow more efficient forwarding, less stringent limits on the length of options, and greater flexibility for introducing new options in the future. The extension headers are: Hop-by-Hop Option, Routing (Type 0), Fragment, Destination Option, Authentication, Encapsulation Payload.

Flow labeling capability - A new capability has been added to enable the labeling of packets belonging to particular traffic flows for which the sender requests special handling, such as non-default Quality of Service or real-time service.

The IPv6 header structure is as follows:



IPv6 header structure

Version
Internet Protocol Version number (IPv6 is 6).

Priority
Enables a source to identify the desired delivery priority of the packets. Priority values are divided into ranges: traffic where the source provides congestion control and non-congestion control traffic.

Flow label
Used by a source to label those products for which it requests special handling by the IPv6 router. The flow is uniquely identified by the combination of a source address and a non-zero flow label.

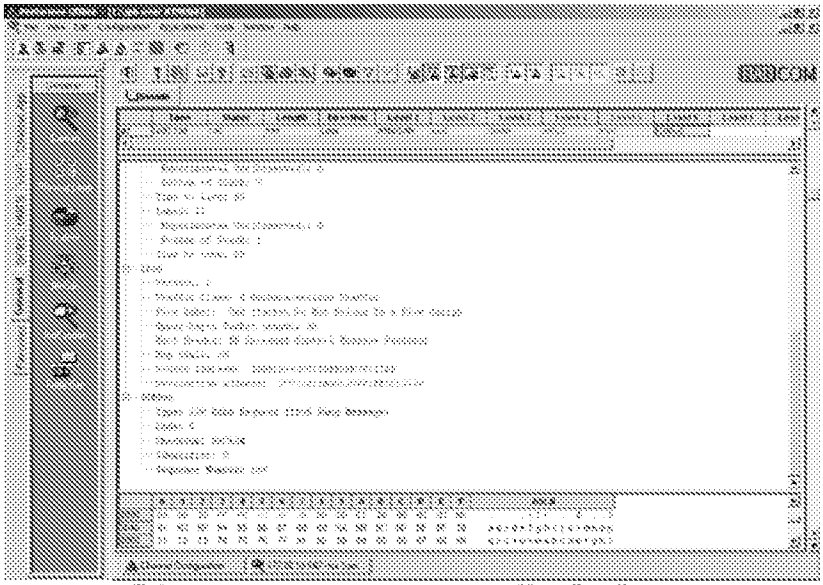
Payload length
Length of payload (in octets).

Next header
Identifies the type of header immediately following the IPv6 header.

Hop limit
8-bit integer that is decremented by one by each node that forwards the packet. The packet is discarded if the Hop Limit is decremented to zero.

Source address
128-bit address of the originator of the packet.

Destination address
128-bit address of the intended recipient of the packet.



Enlarge More Details

Interested in more details about testing this protocol? Click here

TCP

RFC793 http://www.cis.ohio-state.edu/htbin/rfc/rfc793.html
RFC1146 http://www.cis.ohio-state.edu/htbin/rfc/rfc1146.html
RFC1072 http://www.cis.ohio-state.edu/htbin/rfc/rfc1072.html
This RFC has been replaced by RFC 1323.
The information on this page will be updated to suit the new RFC in the near future.
RFC1693 http://www.cis.ohio-state.edu/htbin/rfc/rfc1693.html

IETF RFC793 defines the Transmission Control Protocol (TCP). TCP provides a reliable stream delivery and virtual connection service to applications through the use of sequenced acknowledgment with retransmission of packets when necessary.

The TCP header structure is as follows:

<table>
<tr><td colspan="8">16</td><td>32 bits</td></tr>
<tr><td colspan="8">Source port</td><td>Destination port</td></tr>
<tr><td colspan="9">Sequence number</td></tr>
<tr><td colspan="9">Acknowledgement number</td></tr>
<tr><td>Offset</td><td>Resrvd</td><td>U</td><td>A</td><td>P</td><td>R</td><td>S</td><td>F</td><td>Window</td></tr>
<tr><td colspan="8">Checksum</td><td>Urgent pointer</td></tr>
<tr><td colspan="9">Option + Padding</td></tr>
<tr><td colspan="9">Data</td></tr>
<tr><td colspan="9">TCP header structure</td></tr>
</table>

Source port
Source port number.

Destination port
Destination port number.

Sequence number
The sequence number of the first data octet in this segment (except when SYN is present). If SYN is present, the sequence number is the initial sequence number (ISN) and the first data octet is ISN+1.

Acknowledgment number

If the ACK control bit is set, this field contains the value of the next sequence number which the sender of the segment is expecting to receive. Once a connection is established, this value is always sent.

Data offset
4 bits. The number of 32-bit words in the TCP header, which indicates where the data begins. The TCP header (even one including options) has a length which is an integral number of 32 bits.

Reserved
6 bits. Reserved for future use. Must be zero.

Control bits
6 bits. The control bits may be (from right to left):

U (URG)	Urgent pointer field significant.
A (ACK)	Acknowledgment field significant.
P (PSH)	Push function.
R (RST)	Reset the connection.
S (SYN)	Synchronize sequence numbers.
F (FIN)	No more data from sender.

Window
16 bits. The number of data octets which the sender of this segment is willing to accept, beginning with the octet indicated in the acknowledgment field.

Checksum
16 bits. The checksum field is the 16 bit one's complement of the one's complement sum of all 16-bit words in the header and text. If a segment contains an odd number of header and text octets to be checksummed, the last octet is padded on the right with zeros to form a 16-bit word for checksum purposes. The pad is not transmitted as part of the segment. While computing the checksum, the checksum field itself is replaced with zeros.

Urgent Pointer
16 bits. This field communicates the current value of the urgent pointer as a positive offset from the sequence number in this segment. The urgent pointer points to the sequence number of the octet following the urgent data. This field can only be interpreted in segments for which the URG control bit has been set.

Options
Options may be transmitted at the end of the TCP header and always have a length which is a multiple of 8 bits. All options are included in the checksum. An option may begin on any octet boundary.

There are two possible formats for an option:

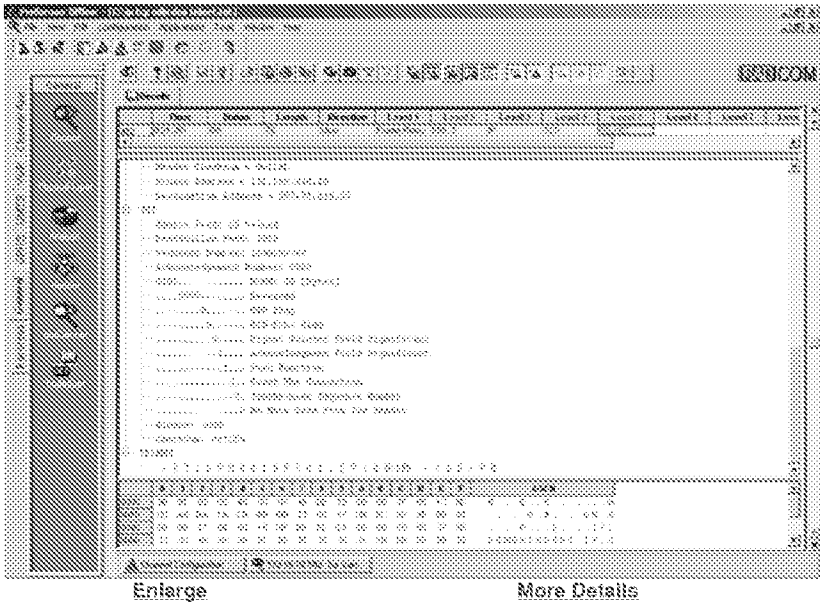
- A single octet of option type.
- An octet of option type, an octet of option length, and the actual option data octets.

The option length includes the option type and option length, as well as the option data octets.

The list of options may be shorter than that designated by the data offset field because the contents of the header beyond the End-of-Option option must be header padding i.e., zero.

A TCP must implement all options.

Data
TCP data or higher layer protocol.



Enlarge More Details

Interested in more details about testing this protocol? Click here

UDP

RFC768 http://www.cis.ohio-state.edu/htbin/rfc/rfc768.html

The User Datagram Protocol (UDP), defined by IETF RFC768, provides a simple, but unreliable message service for transaction-oriented services. Each UDP header carries both a source port identifier and destination port identifier, allowing high-level protocols to target specific applications and services among hosts.

The UDP header structure is shown as follows:

16	32 bits
Source port	Destination port
Length	Checksum
Data	

UDP header structure

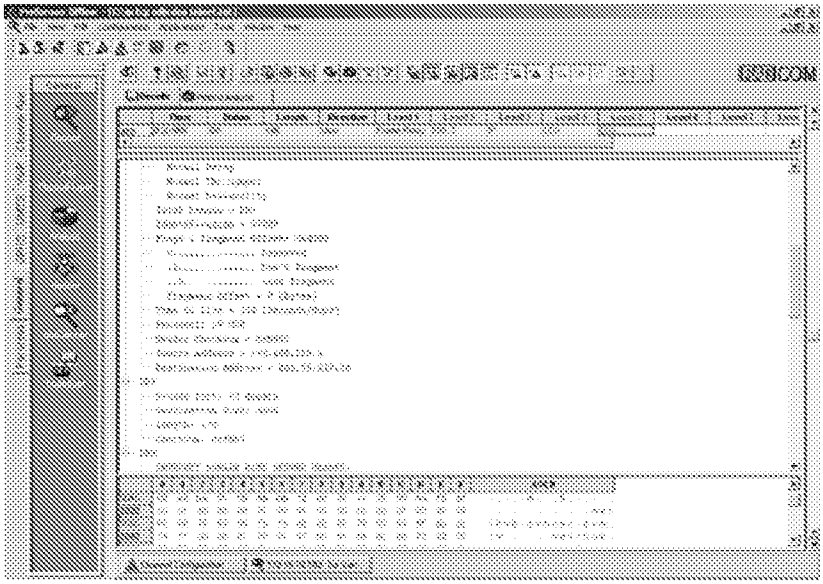
Source port
Source port is an optional field. When used, it indicates the port of the sending process and may be assumed to be the port to which a reply should be addressed in the absence of any other information. If not used, a value of zero is inserted.

Destination port
Destination port has a meaning within the context of a particular Internet destination address.

Length
The length in octets of this user datagram, including this header and the data. The minimum value of the length is eight.

Checksum
The 16-bit one's complement of the one's complement sum of a pseudo header of information from the IP header, the UDP header and the data, padded with zero octets at the end (if necessary) to make a multiple of two octets.

Data
UDP data field.



Enlarge More Details

Interested in more details about testing this protocol? Click here

1 2 3 4 5 6 7 8 9

TCP/IP Family Protocol Information

AH | ATMP | ARP/RARP | BGMP | BGP-4 | COPS | DCAP | DHCP | Diameter | DIS | DNS | DVMRP | EGP | EIGRP | ESP | FANP | Finger | FTP | HSRP | HTTP | ICMP/ICMPv6 | IGMP | IGRP | IMAP4 | IMPPpre/IMPPmes | IPDC | IP | IPv6 | IRC | ISAKMP | ISAKMP/IKE | iSCSI | ISTP | ISP | LDAP | L2F | L2TP | MARS | Mobile IP | MZAP | NARP | NetBIOS/IP | NHRP | NTP | OSPF | PIM | POP3 | PPTP | Radius | RLOGIN | RIP2 | RIPng for IPv6 | RSVP | RTSP | RUDP | SCTP | S-HTTP | SLP | SMTP | SNMP | SOCKS | TACACS+ | TALI | TCP | TELNET | TFTP | TLS | TRIP | UDP | Van Jacobson | VRRP | WCCP | X-Window | XOT

Additional Information

- Protocol Testing - Products
- Protocol Testing - Solutions
- Protocol Testing - Technologies

ATTACHMENT D

Network address translation

From Wikipedia, the free encyclopedia

In computer networking, **network address translation** (NAT, also known as *network masquerading*, *native address translation* or *IP masquerading*) is a technique of transceiving network traffic through a router that involves re-writing the source and/or destination IP addresses and usually also the TCP/UDP port numbers of IP packets as they pass through. Checksums (both IP and TCP/UDP) must also be rewritten to take account of the changes. Most systems using NAT do so in order to enable multiple hosts on a private network to access the Internet using a single public IP address (see gateway). Nonetheless, NAT can introduce complications in communication between hosts and may have a performance impact.

Contents

Overview

NAT first became popular as a way to deal with the IPv4 address shortage and to avoid all the difficulty of reserving IP addresses. NAT has proven particularly popular in countries other than the United States, which (for historical reasons) have fewer address-blocks allocated per capita. It has become a standard feature in routers for home and small-office Internet connections, where the price of extra IP addresses would often outweigh the benefits. NAT also adds to security as it disguises the internal network's structure: all traffic appears to outside parties as if it originates from the gateway machine.

In a typical configuration, a local network uses one of the designated "private" IP address subnets (the RFC 1918 Private Network Addresses are 192.168.x.x, 172.16.x.x through 172.31.x.x, and 10.x.x.x - using CIDR notation, 192.168/16, 172.16/12, and 10/8), and a router on that network has a private address (such as 192.168.0.1) in that address space. The router is also connected to the Internet with a single "public" address (known as "overloaded" NAT) or multiple "public" addresses assigned by an ISP. As traffic passes from the local network to the Internet, the source address in each packet is translated on the fly from the private addresses to the public address(es). The router tracks basic data about each active connection (particularly the destination address and port). When a reply returns to the router, it uses the connection tracking data it stored during the outbound phase to determine where on the internal network to forward the reply; the TCP or UDP client port numbers are used to demultiplex the packets in the case of overloaded NAT, or IP address and port number when multiple public addresses are available, on packet return. To a system on the Internet, the router itself appears to be the source/destination for this traffic.

It has been argued that the wide adoption of IPv6 would make NAT unnecessary, as NAT is a method of handling the shortage of IPv4 address space.

Drawbacks

Hosts behind NAT-enabled routers do not have true end-to-end connectivity and cannot participate in some Internet protocols. Services that require the initiation of TCP connections from the outside network, or stateless protocols such as those using UDP, can be disrupted. Unless the NAT router makes a specific effort to support such protocols, incoming packets cannot reach their destination. Some protocols can accommodate one instance of NAT between participating hosts ("passive mode" FTP, for example), sometimes with the assistance of an Application Layer Gateway (see below), but fail when both systems are separated from the Internet by NAT. Use of NAT also complicates tunneling protocols such as IPsec because NAT modifies values in the headers which interfere with the integrity checks done by IPsec and other tunneling protocols.

End-to-end connectivity has been a core principle of the Internet, supported for example by the Internet Architecture Board. Current Internet architectural documents observe that NAT is a violation of the End-to-End Principle, but that NAT does have a valid role in careful design.[1] There is considerably more concern with the use of IPv6 NAT, and many IPv6 architects believe IPv6 was intended to remove the need for NAT.[2]

Some Internet service providers (ISPs) only provide their customers with "local" IP addresses.Thus, these customers must access services external to the ISP's network through NAT. As a result, it may be argued that such companies do not properly provide "Internet" service.

Benefits

In addition to the convenience and low cost of NAT, the lack of full bidirectional connectivity can be regarded in some situations as a feature rather than a limitation. To the extent that NAT depends on a machine on the local network to initiate any connection to hosts on the other side of the router, it prevents malicious activity initiated by outside hosts from reaching those local hosts. This can enhance the reliability of local systems by stopping worms and enhance privacy by discouraging scans. Many NAT-enabled firewalls use this as the core of the protection they provide.

The greatest benefit of NAT is that it is a practical solution to the impending exhaustion of IPv4 address space. Networks that previously required a Class B IP range or a block of Class C network addresses can now be connected to the Internet with as little as a single IP address (many home networks are set up this way). The more common arrangement is having machines that require true bidirectional and unfettered connectivity supplied with a 'real' IP address, while having machines that do not provide services to outside users tucked away behind NAT with only a few IP addresses used to enable Internet access.

Basic NAT and PAT

Two kinds of network address translation exist:

- **PAT (Port Address Translation)** - The type popularly, but incorrectly, called simply "NAT" (also sometimes named "Network Address Port Translation, NAPT") refers to network address translation involving the mapping of port numbers, allowing multiple machines to share a single IP address.
- **Basic NAT** - The other, technically simpler, forms—"one-to-one NAT", "basic NAT", "static NAT" and "pooled NAT"—involve only address translation, not port mapping. This requires an external IP address for each simultaneous connection. Broadband routers often use this feature, sometimes labelled "DMZ host", to allow a designated computer to accept all external connections even when the router itself uses the only available external IP address.

NAT with port-translation (i.e. PAT) comes in two sub-types: source address translation (**source NAT**), which re-writes the IP address of the computer which initiated the connection; and its counterpart, destination address translation (**destination NAT**). In practice, both are usually used together in coordination for two-way communication.

NOTE: 'PAT' as it is referred to here is referred to by Cisco as NAT 'overloading' as described in this Howstuffworks article provided to Howstuffworks by Cisco: http://computer.howstuffworks.com/nat3.htm

Relationship between NAT and PAT

PAT is closely related to NAT.

In NAT, generally only the IP addresses are modified: There is a 1:1 correspondence between publicly exposed IP addresses and privately held IP addresses. In PAT, both the sender's private IP and port number are modified; the PAT device chooses the port numbers that will be seen by hosts on the public network.

In NAT, incoming packets are routed to their destination IP address on the private network by reference to the incoming source IP address given by the host on the public network. In PAT, there is generally only one publicly exposed IP address and incoming packets from the public network are routed to their destinations on the private network by reference to a table held within the PAT device that keeps track of public and private port pairs. This is often called connection tracking.

Some devices that claim to offer NAT, such as broadband routers, actually offer PAT. For this reason, there is considerable confusion between the terms. The common use of NAT to include PAT devices suggests that PAT should be considered a type of NAT rather than a distinct technology.

NAT and TCP/UDP

"Pure NAT", operating on IP alone, may or may not correctly pass protocols that are totally concerned with IP information, such as ICMP, depending on whether the payload is interpreted by a host on the "inside" or "outside" of translation. As soon as the protocol stack is climbed, even with such basic protocols such TCP and UDP, the protocols will break unless NAT takes action beyond the network layer.

IP has a checksum in each packet header, which provides error detection only for the header. IP datagrams may become fragmented and it is necessary for a NAT to reassemble these fragments to allow correct recalculation of higher level checksums and correct tracking of which packets belong to which connection.

The major transport layer protocols, TCP and UDP, have a checksum that covers all the data they carry, as well as the TCP/UDP header, plus a "pseudo-header" that contains the source and destination IP addresses of the packet carrying the TCP/UDP header. For an originating NAT to successfully pass TCP or UDP, it must recompute the TCP/UDP header checksum based on the translated IP addresses, not the original ones, and put that checksum into the TCP/UDP header of the first packet of the fragmented set of packets. The receiving NAT must recompute the IP checksum on every packet it passes to the destination host, and also recognize and recompute the TCP/UDP header using the retranslated addresses and pseudo-header. This is not a completely solved problem. One solution is for the receiving NAT to reassemble the entire segment and then recompute a checksum calculated across all packets.

It may be wise for the originating host to do MTU Path Discovery (RFC1191) to determine what MTU will go to the end without fragmentation, and then set the "don't fragment" bit in the appropriate packets. There is no totally general solution to this problem, which is why one of the goals of IPv6 is to avoid NAT.

Applications affected by NAT

Some higher-layer protocols (such as FTP and SIP) send network layer address information inside application payloads. FTP in active mode, for example, uses separate connections for control traffic (commands) and for data traffic (file contents). When requesting a file transfer, the host making the request identifies the corresponding data connection by its network layer and transport layer addresses. If the host making the request lies behind a simple NAT firewall, the translation of the IP address and/or TCP port number makes the information received by the server invalid.

An Application Layer Gateway (ALG) can fix this problem. An ALG software module running on a NAT firewall device updates any payload data made invalid by address translation. ALGs obviously need to understand the higher-layer protocol that they need to fix, and so each protocol with this problem requires a separate ALG.

Another possible solution to this problem is to use NAT traversal techniques using protocols such as STUN or ICE or proprietary approaches in a session border controller. NAT traversal is possible in both TCP- and UDP-based applications, but the UDP-based technique is simpler, more widely understood, and more compatible with legacy NATs. In either case, the high level protocol must be designed with NAT traversal in mind, and it does not work reliably across symmetric NATs or other poorly-behaved legacy NATs.

Other possibilities are UPnP (Universal Plug and Play) or Bonjour (NAT-PMP), but these require the cooperation of the NAT device.

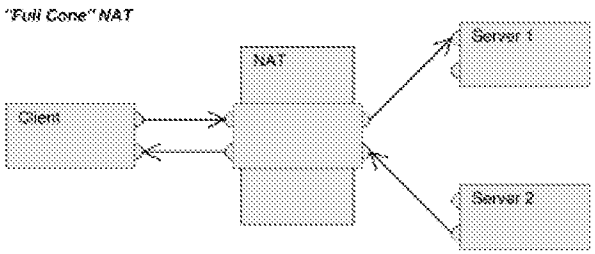
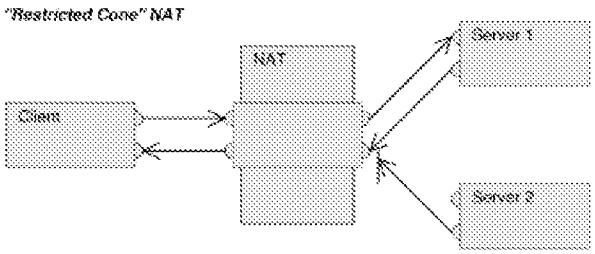
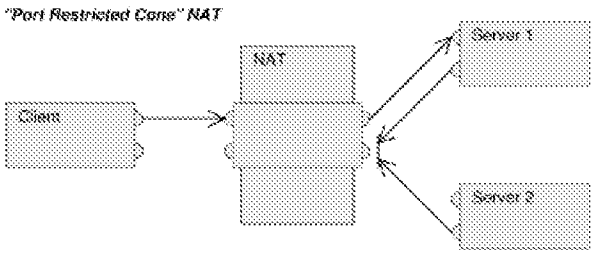
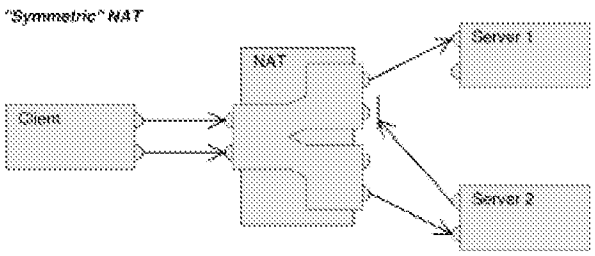
Most traditional client-server protocols (FTP being the main exception), however, do not send layer 3 contact information and therefore do not require any special treatment by NATs. In fact, avoiding NAT complications is practically a requirement when designing new higher-layer protocols today.

NATs can also cause problems where IPsec encryption is applied and in cases where multiple devices such as SIP phones are located behind a NAT. Phones which encrypt their signalling with IPsec encapsulate the port information within the IPsec packet meaning that NA(P)T devices cannot access and translate the port. In these cases the NA(P)T devices revert to simple NAT operation. This means that all traffic returning to the NAT will be mapped onto one client causing the service to fail. There are a couple of solutions to this problem, one is to use TLS which operates at level 4 in the OSI Reference Model and therefore does not mask the port number, or to Encapsulate the IPsec within UDP - the latter being the solution chosen by TISPAN to achieve secure NAT traversal.

It is also a problem with Xbox Live gameplay when the need for players to communicate becomes harder and slows down gameplay.

Different types of NAT

Applications that deal with NAT sometimes need to characterize NAT by type. The STUN protocol proposed to characterize Network address translation as **Full cone NAT**, **restricted cone NAT**, **port restricted cone NAT** or **symmetric NAT**.[3][4]

Full cone NAT, also known as one-to-one NAT ▪ Once an internal address (iAddr:port1) is mapped to an external address (eAddr:port2), any packets from iAddr:port1 will be sent through eAddr:port2. Any external host can send packets to iAddr:port1 by sending packets to eAddr:port2.	
Restricted cone NAT ▪ Once an internal address (iAddr:port1) is mapped to an external address (eAddr:port2), any packets from iAddr:port1 will be sent through eAddr:port2. An external host (hostAddr:any) can send packets to iAddr:port1 by sending packets to eAddr:port2 only if iAddr:port1 had previously sent a packet to hostAddr:any. "any" means the port number doesn't matter.	
Port restricted cone NAT Like a **restricted cone NAT**, but the restriction includes port numbers. ▪ Once an internal address (iAddr:port1) is mapped to an external address (eAddr:port2), any packets from iAddr:port1 will be sent through eAddr:port2. An external host (hostAddr:port3) can send packets to iAddr:port1 by sending packets to eAddr:port2 only if iAddr:port1 had previously sent a packet to hostAddr:port3.	
Symmetric NAT ▪ Each request from the same internal IP address and port to a specific destination IP address and port is mapped to a unique external source IP address and port. If the same internal host sends a packet even with the same source address and port but to a different destination, a different mapping is used. ▪ Only an external host that receives a packet from an internal host can send a packet back.	

This terminology has been the source of much confusion, as it has proven inadequate at describing real-life NAT behavior.[5] Many NAT implementations combine the specified types, and it is therefore better to refer to specific individual NAT behaviors instead of using the Cone/Symmetric terminology. Especially, most NAT translators combine *symmetric NAT* for outgoing connections with a *static port mapping* capability. The latter means that all incoming packets to the specific external address and port can be redirected to a specific internal address and port. Some products can redirect packets to several internal hosts - e.g. to divide the load between a few servers (however, this introduces problems with more sophisticated communications having many interconnected packets and thus is rarely used.

Many NAT implementations follow a **port preservation** design. For most communications, they will use the same values as internal and external port numbers. However, if two internal hosts attempt to communicate with the same external host using the same port number, the external port number used by the second host will be chosen at random. Such NAT will be sometimes perceived as **restricted cone NAT** and other times as **symmetric NAT**.

Other examples of use

- Load Balancing: Destination NAT can redirect connections pointed at some server to randomly selected servers.
- Failover: Destination NAT can be used to set up a service requiring high availability. If a system involves a critical server accessed through a router, and if the router detects that server has gone down, it could use destination NAT to transparently re-route a connection to arrive on a backup server.
- Transparent proxying: NAT can redirect HTTP connections targeted at the Internet to a special HTTP proxy which can cache content and filter requests. Some internet service providers use this technique to reduce bandwidth usage without requiring their clients to configure their web browser for proxy support.
- Overlapping Networks: Advanced NAT configurations can connect two networks that have overlapping addresses, such as Private addresses, and this is a very useful feature for companies that have just started to merge their networks. This requires that both Source & Destination IP addresses be replaced at the same time, fooling each other's networks.

See also

- Port address translation (PAT, can be used in conjunction with NAT)
- Firewall (networking)
- Proxy server
- Middlebox
- Routing
- Gateway (telecommunications)
- Subnet
- IPv6
- UDP hole punching
- AYIYA (IPv6 over IPv4 UDP thus working IPv6 tunneling over most NATs)
- IPv4 address exhaustion
- Private IP address
- Internet Connection Sharing
- Port forwarding
- STUN: Simple Traversal of UDP over NATs
- Teredo tunneling: NAT traversal using IPv6
- Internet Gateway Device (IGD) Protocol: UPnP NAT-traversal method
- NAT-PT

Popular NAT software

- IPFilter
- PF (firewall): The OpenBSD Packet Filter.
- Iptables masquerading
- Berkeley Software Distribution
- Internet Connection Sharing (ICS)
- WinGate
- Cisco IOS

References

1. ^ Some Internet Architectural Guidelines and Philosophy,RFC 3439, R. Bush & D. Meyer,December 2002
2. ^ Local Network Protection for IPv6,RFC 4864, G. Van de Velde *et al.*,May 2007
3. ^ STUN
4. ^ NAT Types (PDF).
5. ^ Francois Audet, Cullen Jennings (January 2007). "*RFC 4787 Network Address Translation (NAT) Behavioral Requirements for Unicast UDP*" (text). IETF. Retrieved on 2007-08-29.

External links

- TeleDict: Introduction to NAT
- Characterization of different TCP NATs – Paper discussing the different types of NAT
- P2P & NAT connections (In Spanish PDF)
- Test the NAT type of your routerA small tool by the eMule developers to detect the nat type of your router.
- Anatomy: A Look Inside Network Address Translators – Volume 7, Issue 3, September 2004
- HowStuffWorks: *How Network Address Translation Works* by Jeff Tyson
- NAT traversal techniques in multimedia Networks – White Paper from Newport Networks

- Peer-to-Peer Communication Across Network Address Translators (PDF) – NAT traversal techniques for UDP and TCP
- RFC 4008 – Standards Track – Definitions of Managed Objects for Network Address Translators (NAT)
- RFC 3022 – Traditional IP Network Address Translator (Traditional NAT)
- RFC 1631 – Obsolete – The IP Network Address Translator (NAT)
- nat-traverse – Tool to establish tunnels through NAT gateways without need for reconfiguration of the involved routers
- Speak Freely End of Life Announcement – John Walker's discussion of why he stopped developing a famous program for free internet communication, part of which is directly related to NAT.
- NATs, IPsec and VoIP – White paper looks at the issues of IPsec and NAT devices and how UDP encapsulation of IPsec solves the problems.
- Cisco IOS NAT Application Layer Gateway
- natd
- A blog entry explanation of NAT
- Alternative Taxonomy
  - Static NAT
  - Dynamic NAT
  - Masquerade NAT

Retrieved from "http://en.wikipedia.org/wiki/Network_address_translation"
Categories: Network Address Translation
Hidden categories: All articles with unsourced statements | Articles with unsourced statements since March 2008 | Misleading articles